

Unbeatable Consensus*

Armando Castañeda[†]

Yannai A. Gonczarowski[‡]

Yoram Moses[§]

September 15, 2014

Abstract

The *unbeatability* of a consensus protocol, introduced by Halpern, Moses and Waarts in [14], is a stronger notion of optimality than the accepted notion of early stopping protocols. Using a novel knowledge-based analysis, this paper derives the first practical unbeatable consensus protocols in the literature, for the standard synchronous message-passing model with crash failures. These protocols strictly dominate the best known protocols for uniform and for non-uniform consensus, in some case beating them by a large margin. The analysis provides a new understanding of the logical structure of consensus, and of the distinction between uniform and nonuniform consensus. Finally, the first (early stopping and) unbeatable protocol that treats decision values “fairly” is presented. All of these protocols have very concise descriptions, and are shown to be efficiently implementable.

Keywords: Consensus, uniform consensus, optimality, knowledge

1 Introduction

Following [15], we say that a protocol P is a *worst-case optimal* solution to a decision task S in a given model if it solves S , and decisions in P are always taken no later than the *worst-case* lower bound for decisions in this problem, in that model. Here we consider standard synchronous message-passing models with n processes and at most $t < n$ crash failures per run; it will be convenient to denote the number of *actual* failures in a given run by f . Processes proceed in a sequence of synchronous rounds. The very first consensus protocols were worst-case optimal, deciding in exactly $t + 1$ rounds in all runs [6, 19]. It was soon realized, however, that they could be strictly improved upon by *early stopping* protocols [5], which are also worst-case optimal, but can often decide much faster than the original ones. This paper presents a number of consensus protocols that are not only worst-case optimal and early stopping, but furthermore cannot be strictly improved upon, and are thus optimal in a much stronger sense.

In benign failure models it is typically possible to define the behaviour of the environment (i.e., the adversary) in a manner that is independent of the protocol, in terms of a pair $\alpha = (\vec{v}, F)$ consisting of a vector \vec{v} of initial values and a failure pattern F . (A formal definition is given in Section 2.) A failure model \mathcal{F} is identified with a set of (possible) failure patterns. For ease of exposition, we will think of such a pair $\alpha = (\vec{v}, F)$ as a particular *adversary*. In a synchronous environment, a deterministic protocol P and an adversary α uniquely define a run $r = P[\alpha]$. With this terminology, we can compare the performance of different decision protocols solving a particular

*Part of the results of this paper were announced in [1].

[†]Universidad Nacional Autónoma de México (UNAM), *E-mail*: armando@cs.technion.ac.il.

[‡]The Hebrew University of Jerusalem and Microsoft Research, *E-mail*: yannai@gonch.name.

[§]Technion, *E-mail*: moyses@ee.technion.ac.il.

task in a given context $\gamma = (\vec{V}, \mathcal{F})$, where \vec{V} is a set of possible vectors of initial values. A decision protocol Q **dominates** a protocol P in γ , denoted by $Q \preceq_{\gamma} P$ if, for all adversaries α and every process i , if i decides in $P[\alpha]$ at time m_i , then i decides in $Q[\alpha]$ at some time $m'_i \leq m_i$. Moreover, we say that Q **strictly dominates** P if $Q \preceq_{\gamma} P$ and $P \not\preceq_{\gamma} Q$. I.e., if Q dominates P and for some $\alpha \in \gamma$ there exists a process i that decides in $Q[\alpha]$ *strictly before* it does so in $P[\alpha]$. In the crash failure model, the early-stopping protocols of [5] strictly dominate the original protocols of [19], in which decisions are always performed at time $t + 1$. Nevertheless, these early stopping protocols may not be optimal solutions to consensus. Following [15] a protocol P is said to be an **all-case optimal** solution to a decision task S in a context γ if it solves S and, moreover, it dominates every protocol P' that solves S in γ . Dwork and Moses presented all-case optimal solutions to the *simultaneous* variant of consensus [8]. For the standard (*eventual*) variant of consensus, in which decisions are not required to occur simultaneously, Moses and Tuttle showed that no all-case optimal solution exists [18]. Consequently, Halpern, Moses and Waarts in [14] initiated the study of a natural notion of optimality that is achievable by eventual consensus protocols:

Definition 1 (Halpern, Moses and Waarts). *A protocol P is an **unbeatable** solution to a decision task S in a context γ if P solves S in γ and no protocol Q solving S in γ strictly dominates P .¹*

Halpern, Moses and Waarts observed that for every consensus protocol P there exists an unbeatable protocol Q_P that dominates P . Moreover, they showed a two-step transformation that defines such a protocol Q_P based on P . This transformation and the resulting protocols are based on a notion of *continual* common knowledge that is computable, but not efficiently: in the resulting protocol, each process executes exponential time (PSPACE) local computations in every round. The logical transformation is not applied in [14] to an actual protocol. As an example of an unbeatable protocol, they present a particular protocol, called $P0_{\text{opt}}$, and argue that it is unbeatable in the crash failure model. Unfortunately, as we will show, $P0_{\text{opt}}$ is in fact beatable. This does not refute the general analysis and transformation defined in [14]; they remain correct. Rather, the fault is in an unsound step in the proof of optimality of $P0_{\text{opt}}$ (Theorem 6.2 of [14]), in which an inductive step is not explicitly detailed, and does not hold.

The main contributions of this paper are:

1. A knowledge-based analysis is applied to the classical consensus protocol, and is shown to yield solutions that are optimal in a much stronger sense than all previous solutions. Much simpler and more intuitive than the framework used in [14], it illustrates how the knowledge-based approach can yield a structured approach to the derivation of efficient protocols.
2. OPT_0 , the first explicit unbeatable protocol for nonuniform consensus is presented. It is computationally efficient, and its unbeatability is established by way of a succinct proof. Moreover, OPT_0 is shown to strictly dominate the $P0_{\text{opt}}$ protocol from [14], proving that the latter is in fact beatable.
3. An analysis of uniform consensus gives rise to U-OPT_0 , the first explicit unbeatable protocol for uniform consensus. The analysis used in the design of U-OPT_0 sheds light on the inherent difference and similarities between the uniform and nonuniform variants of consensus in this model.

¹All-case optimal protocols are called “*optimal in all runs*” in [8]. They are termed “*optimum*” in [14], while unbeatable protocols are simply called “*optimal*” there. We prefer the term *unbeatable* because “optimal” is used very broadly, and inconsistently, in the literature.

4. Early stopping protocols for consensus are traditionally one-sided, preferring to decide on 0 (or on 1) if possible. deciding on a predetermined value (say, 0) if possible, we present an unbeatable (and early stopping) majority consensus protocol OPT_{Maj} is presented, that prefers the *majority* value.
5. We identify the notion of a *hidden path* as being crucial to decision in the consensus task. If a process identifies that no hidden path exists, then it can decide. In the fastest early-stopping protocols, a process decides after the first round in which it does not detect a new failure. By deciding based on the nonexistence of a hidden path, our unbeatable protocols can stop up to $t - 3$ rounds faster than the best early stopping protocols in the literature.

We now sketch the intuition behind, our unbeatable consensus protocols.

In the standard version of consensus, every process i starts with an initial value $v_i \in \{0, 1\}$, and the following properties must hold in every run r :

(Nonuniform) Consensus:

Decision: Every correct process must decide on some value,

Validity: If all initial values are v then the correct processes decide v , and

Agreement: All correct processes decide on the same value.

The connection between knowledge and distributed computing was proposed in [13] and has been used in the analysis of a variety of problems, including consensus (see [9] for more details and references). In this paper, we employ simpler techniques to perform a more direct knowledge-based analysis. Our approach is based on a simple principle recently formulated by Moses in [17], called the **knowledge of preconditions** principle (**KoP**), which captures an essential connection between knowledge and action in distributed and multi-agent systems. Roughly speaking, the **KoP** principle says that if C is a necessary condition for an action α to be performed by process i , then $K_i(C)$ — i knowing C — is a necessary condition for i performing α . E.g., it is not enough for a client to have positive credit in order to receive cash from an ATM; the ATM must **know** that the client has positive credit.

Problem specifications typically state or imply a variety of necessary conditions. In the crash failure model studied in this paper, we will say that a process is **active** at time m in a given run, if it does not crash before time m . For $v \in \{0, 1\}$, we denote by $\text{decide}_i(v)$ the action of i deciding v , and use \bar{v} as shorthand for $1 - v$.

Lemma 1. *Consensus implies the following necessary conditions for $\text{decide}_i(v)$ in the crash failure model:*

- (a) “at least one processes had initial value v ” (we denote this by $\exists v$), and
- (b) “no currently active process has decided, or is currently deciding, \bar{v} ” (we denote this by $\text{no-decided}(\bar{v})$).

Both parts follow from observing that if i decides v at a point where either (a) or (b) does not hold, then the execution can be extended to a run in which i (as well as j , for (b)) is correct (does not crash), and this run violates **Validity** for (a) or **Agreement** for (b).

Given Lemma 1, **KoP** implies that $K_i \exists v$ and $K_i \text{no-decided}(\bar{v})$ are also necessary conditions for $\text{decide}_i(v)$. In this paper, we will explore how this insight can be exploited in order to design efficient consensus protocols. Indeed, our first unbeatable protocol will be one in which, roughly

speaking, the rule for $\text{decide}_i(0)$ will be $K_i \exists 0$, and the rule for $\text{decide}_i(1)$ will be $K_i \text{no-decided}(0)$. As we will show, if the rule for $\text{decide}_i(0)$ is $K_i \exists 0$, then $\text{no-decided}(0)$ reduces to the fact $\text{not-known}(\exists 0)$, which is true at a given time if $K_j \exists 0$ holds for no currently-active process j . Thus, $K_i \text{no-decided}(0)$ — our candidate rule for deciding 1 — then becomes $K_i \text{not-known}(\exists 0)$. While $K_i \exists 0$ involves the knowledge a process has about initial values, $K_i \text{not-known}(\exists 0)$ is concerned with i 's knowledge about the knowledge of others. We will review the formal definition of knowledge in the next section, in order to turn this into a rigorous condition.

Converting the above description into an actual protocol essentially amounts to providing concrete tests for when these knowledge conditions hold. It is straightforward to show (and quite intuitive) that in a full-information protocol $K_i \exists 0$ holds exactly if there is a message chain from some process j whose initial value is 0, to process i . To determine that $\text{not-known}(\exists 0)$, a process must have proof that no such chain can exist. Our technical analysis identifies a notion of a *hidden path* with respect to i at a time m , which implies that a message chain could potentially be communicating a value unbeknownst to i . It is shown that hidden paths are key to evaluating whether $K_i \text{not-known}(\exists 0)$ holds. In fact, it turns out that hidden paths are key to obtaining additional unbeatable protocols in the crash failure model. We present two such protocols; one is a consensus protocol in which a process that sees a majority value can decide on this value, and the other is an unbeatable protocol for the *uniform* variant of consensus. In uniform consensus, any two processes that decide must decide on the same value, even if one (or both) of them crash soon after deciding.

This paper is structured as follows: The next section reviews the definitions of the synchronous crash-failure model and of knowledge in this model. Section 3 presents OPT_0 , our unbeatable consensus protocol, proves its unbeatability, and shows that it beats the protocol $P0_{\text{opt}}$ of [14]. It then derives an unbeatable consensus protocol, OPT_{Maj} , that treats 0 and 1 in a balanced way. Both unbeatable protocols decide in no more than $f + 1$ rounds in runs in which f processes actually fail but they can decide much earlier than that. Section 4 studies uniform consensus, and derives U-OPT_0 , an unbeatable protocol for uniform consensus. Finally, Section 5 concludes with a discussion. The Appendix contains full proofs to all claims that are not proved in the main text.

2 Preliminary Definitions

Our model of computation is the standard synchronous message-passing model with benign crash failures. A system has $n \geq 2$ processes denoted by $\text{Procs} = \{1, 2, \dots, n\}$. Each pair of processes is connected by a two-way communication link, and each message is tagged with the identity of the sender. They share a discrete global clock that starts out at time 0 and advances by increments of one. Communication in the system proceeds in a sequence of *rounds*, with round $m + 1$ taking place between time m and time $m + 1$. Each process starts in some *initial state* at time 0, usually with an *input value* of some kind. In every round, each process first performs a local computation, and performs local actions, then it sends a set of messages to other processes, and finally receives messages sent to it by other processes during the same round. We consider the local computations and sending actions of round $m + 1$ as being performed at time m , and the messages are received at time $m + 1$.

A faulty process fails by *crashing* in some round $m \geq 1$. It behaves correctly in the first $m - 1$ rounds and sends no messages from round $m + 1$ on. During its crashing round m , the process may succeed in sending messages on an arbitrary subset of its links. At most $t \leq n - 1$ processes fail in any given execution.

It is convenient to consider the state and behaviour of processes at different (process-time) nodes, where a *node* is a pair $\langle i, m \rangle$ referring to process i at time m . A *failure pattern* describes how

processes fail in an execution. It is a layered graph F whose vertices are nodes $\langle i, m \rangle$ for $i \in \text{Procs}$ and $m \geq 0$. Such a vertex denotes process i and time m . An edge has the form $(\langle i, m-1 \rangle, \langle j, m \rangle)$ and it denotes the fact that a message sent by i to j in round m would be delivered successfully. Let $\text{Crash}(t)$ denote the set of failure patterns in which all failures are crash failures, and no more than t crash failures occur. An **input vector** describes the initial values that the processes receive in an execution. The only inputs we consider are initial values that processes obtain at time 0. An input vector is thus a tuple $\vec{v} = (v_1, \dots, v_n)$ where v_j is the input to process j . We think of the input vector and the failure pattern as being determined by an external scheduler, and thus a pair $\alpha = (\vec{v}, F)$ is called an *adversary*.

A **protocol** describes what messages a process sends and what decisions it takes, as a deterministic function of its local state at the start of a round and the messages received during a round. We assume that a protocol P has access to the values of n and t , typically passed to P as parameters.

A **run** is a description of an infinite behaviour of the system. Given a run r and a time m , we denote by $r_i(m)$ the **local state** of process i at time m in r , and the **global state** at time m is defined to be $r(m) = \langle r_1(m), r_2(m), \dots, r_n(m) \rangle$. A protocol P and an adversary α uniquely determine a run, and we write $r = P[\alpha]$.

Since we restrict attention to benign failure models and focus on decision times and solvability in this paper, it is sufficient to consider *full-information* protocols (*fip*'s for short), defined below [3]. There is a convenient way to consider such protocols in our setting. With an adversary $\alpha = (\vec{v}, F)$ we associate a **communication graph** \mathcal{G}_α , consisting of the graph F extended by labelling the initial nodes $\langle j, 0 \rangle$ with the initial states v_j according to α . Every node $\langle i, m \rangle$ is associated with a subgraph $\mathcal{G}_\alpha(i, m)$ of \mathcal{G}_α , which we think of as i 's *view* at $\langle i, m \rangle$. Intuitively, this graph will represent all nodes $\langle j, \ell \rangle$ from which $\langle i, m \rangle$ has heard, and the initial values it has seen. Formally, $\mathcal{G}_\alpha(i, m)$ is defined by induction on m . $\mathcal{G}_\alpha(i, 0)$ consists of the node $\langle i, 0 \rangle$, labelled by the initial value v_i . Assume that $\mathcal{G}_\alpha(1, m), \dots, \mathcal{G}_\alpha(n, m)$ have been defined, and let $J \subseteq \text{Procs}$ be the set of processes j such that $j = i$ or $e_j = (\langle j, m \rangle, \langle i, m+1 \rangle)$ is an edge of F . Then $\mathcal{G}_\alpha(i, m+1)$ consists of the node $\langle i, m+1 \rangle$, the union of all graphs $\mathcal{G}_\alpha(j, m)$ with $j \in J$, and the edges $e_j = (\langle j, m \rangle, \langle i, m+1 \rangle)$ for all $j \in J$. We say that (j, ℓ) is *seen* by $\langle i, m \rangle$ if (j, ℓ) is a node of $\mathcal{G}_\alpha(i, m)$. Note that this occurs exactly if the failure pattern F allows a (Lamport) message chain from $\langle j, \ell \rangle$ to $\langle i, m \rangle$.

A full-information protocol P is one in which at every node $\langle i, m \rangle$ of a run $r = P[\alpha]$ the process i constructs $\mathcal{G}_\alpha(i, m)$ after receiving its round m nodes, and sends $\mathcal{G}_\alpha(i, m)$ to all other processes in round $m+1$. In addition, P specifies what decisions i should take at $\langle i, m \rangle$ based on $\mathcal{G}_\alpha(i, m)$. Full-information protocols thus differ only in the decisions taken at the nodes. Let $d(i, m)$ be status of i 's decision at time m (either ' \perp ' if it is undecided, or a concrete value ' v '). Thus, in a run $r = P[\alpha]$, we define the local state $r_i(m) = \langle d(i, m), \mathcal{G}_\alpha(i, m) \rangle$ if i does not crash before time m according to α , and $r_i(m) = \odot$, an uninformative "crashed" state, if i crashes before time m .

For ease of exposition and analysis, all of our protocols are full-information. However, in fact, they can all be implemented in such a way that any process sends any other process a total of $O(f \log n)$ bits throughout any execution (as shown by Lemma 23 in Appendix A.5).

2.1 Knowledge

Our construction of unbeatable protocols will be assisted and guided by a knowledge-based analysis, in the spirit of [9, 13]. Runs are dynamic objects, changing from one time point to the next. E.g., at one point process i may be undecided, while at the next it may decide on a value. Similarly, the set of initial values that i knows about, or has seen, may change over time. In general, whether a process "knows" something at a given point can depend on what is true in other runs in which the process has the same information. We will therefore consider the truth of facts at *points* (r, m) —

time m in run r , with respect to a set of runs R (which we call a **system**). We will be interested in systems of the form $R_P = R(P, \gamma)$ where P is a protocol and $\gamma = \gamma(\mathbf{v}^n, \mathcal{F})$ is the set of all adversaries that assign initial values from \mathbf{V} and failures according to \mathcal{F} . We will write $(R, r, m) \models A$ to state that fact A holds, or is satisfied, at (r, m) in the system R .

The truth of some facts can be defined directly. For example, the fact $\exists \mathbf{v}$ will hold at (r, m) in R if some process has initial value \mathbf{v} in $(r, 0)$. We say that (satisfaction of) a fact A is **well-defined in R** if for every point (r, m) with $r \in R$ we can determine whether or not $(R, r, m) \models A$. Satisfaction of $\exists \mathbf{v}$ is thus well defined. Moreover, any boolean combination of well-defined facts is also well defined. We will write $K_i A$ to denote that **process i knows A** , and define:

Definition 2 (Knowledge). *Suppose that A is well defined in R . Define that*
 $(R, r, m) \models K_i A \quad \text{iff} \quad (R, r', m) \models A \text{ holds for all } r' \in R \text{ with } r_i(m) = r'_i(m).$

Thus, if A is well defined in R then Definition 2 makes $K_i A$ well defined in R . Note that what a process knows or does not know depends on its local state. The definition can then be applied recursively, to define the truth of $K_j K_i A$ etc. Knowledge has been used to study a variety of problems in distributed computing. In particular, we now formally define $(R, r, m) \models \text{not-known}(\exists 0)$ to hold iff $(R, r, m) \not\models K_j \exists 0$ holds for every process j that does not crash by time m in r . We will make use of the following fundamental connection between knowledge and actions in distributed systems. A fact A is a **necessary condition** for process i performing action σ (e.g. deciding on an output value) in R if $(R, r, m) \models A$ whenever i performs σ at a point (r, m) of R .

Theorem 1 (Knowledge of Preconditions, [17]). *Let $R_P = R(P, \gamma)$ be the set of runs of a deterministic protocol P . If A is a necessary condition for i performing σ in R_P , then so is $K_i A$.*

3 Unbeatable Consensus

We start with the standard version of consensus defined in the Introduction, and consider the crash failure context $\gamma_{\text{cr}}^t = \langle \mathbf{v}^n, \text{Crash}(t) \rangle$, where $\mathbf{V} = \{0, 1\}$ — initial values are binary bits. Every protocol P in this setting determines a system $R_P = R(P, \gamma_{\text{cr}}^t)$. Recall that Lemma 1 establishes necessary conditions for decision in consensus. Based on this, Theorem 1 yields:

Lemma 2. *Let P be a consensus protocol for γ_{cr}^t and let $R_P = R(P, \gamma_{\text{cr}}^t)$. Then both $K_i \exists \mathbf{v}$ and $K_i \text{no-decided}(\bar{\mathbf{v}})$ are necessary conditions for $\text{decide}_i(\mathbf{v})$ in R_P .*

An analysis of knowledge for *fips* in the crash failure model was first performed by Dwork and Moses in [8]. The following result is an immediate consequence of that analysis. Under the full-information protocol, we have:

Lemma 3 (Dwork and Moses [8]). *Let P be a *fip* in γ_{cr}^t and let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$. For all processes i, j , $(R_P, r, t+1) \models K_i \exists \mathbf{v}$ iff $(R_P, r, t+1) \models K_j \exists \mathbf{v}$.*

Of course, a process that does not know $\exists 0$ must itself have an initial value of 1. Hence, based on Lemma 3, it is natural to design a *fip*-based consensus protocol that performs $\text{decide}_i(0)$ at time $t+1$ if $K_i \exists 0$, and otherwise performs $\text{decide}_i(1)$. (In the very first consensus protocols, all decisions are performed at time $t+1$ [19].) Indeed, one can use Lemma 3 to obtain a strictly better protocol, in which decisions on 0 are performed sooner:

Protocol P_0 (for an undecided process i at time m):

if $K_i \exists 0$ **then** $\text{decide}_i(0)$
elseif $m = t+1$ **then** $\text{decide}_i(1)$

Notice that in a *fip* consensus protocol, it is only necessary to describe the rules for $\text{decide}_i(0)$ and $\text{decide}_i(1)$, since in every round a process sends all it knows to all processes. Since $K_i \exists 0$ is a necessary condition for $\text{decide}_i(0)$, the protocol P_0 decides on 0 as soon as any consensus protocol can. In the early 80's Dolev suggested a closely related protocol B (standing for “Beep”) for γ_{cr}^t , in which processes decide 0 and broadcast the existence of a 0 when they see a 0, and decide 1 at $t + 1$ otherwise [4]; for all adversaries, it performs the same decisions at the same times as P_0 . Halpern, Moses and Waarts show in [14] that for every consensus protocol P in γ_{cr}^t there is an unbeatable consensus protocol Q dominating P . Our immediate goal is to obtain an unbeatable consensus protocol dominating P_0 . To this end, we make use of the following.

Lemma 4. *If $Q \preceq P_0$ is a consensus protocol, then $\text{decide}_i(0)$ is performed in Q exactly when $K_i \exists 0$ first holds.*

We can now formalize the discussion in the Introduction, showing that if decisions on 0 are performed precisely when $K_i \exists 0$ first holds, then $\text{no-decided}(0)$ reduces to $\text{not-known}(\exists 0)$.

Lemma 5. *Let P be a *fip*, in which $\text{decide}_i(0)$ is performed in P exactly when $K_i \exists 0$ first holds, and let $R_P = R(P, \gamma_{\text{cr}}^t)$. Then $(R_P, r, m) \models K_i \text{no-decided}(0)$ iff $(R_P, r, m) \models K_i \text{not-known}(\exists 0)$ for all $r \in R_P$ and $m \geq 0$.*

The proof of Lemma 5 is fairly immediate: If $(R_P, r, m) \not\models K_i \text{not-known}(\exists 0)$ then there is a run r' of R_P such that both $r_i(m) = r'_i(m)$ and $(R_P, r', m) \models K_j \exists 0$ for some correct process j ; therefore, process j decides 0 in r' . The other direction follows directly from the decision rule for 0. We can now define a *fip* consensus protocol in which 0 is decided as soon as its necessary condition $K_i \exists 0$ holds, and 1 is decided as soon as possible, given the rule for deciding 0:

Protocol OPT_0 (for an undecided process i at time m):

if	$K_i \exists 0$	then	$\text{decide}_i(0)$
elseif	$K_i \text{not-known}(\exists 0)$	then	$\text{decide}_i(1)$

We can show that OPT_0 is, indeed, an unbeatable protocol:

Theorem 2. *OPT_0 is an unbeatable consensus protocol in γ_{cr}^t .*

3.1 Testing for Knowing that Nobody Knows

OPT_0 is not a standard protocol, because its actions depend on tests for process i 's knowledge. (It is a *knowledge-based program* in the sense of [9].) In order to turn it into a standard protocol, we need to replace these by explicit tests on the processes' local states. The rule for $\text{decide}_i(0)$ is easy to implement. By Lemma 3(a), $K_i \exists 0$ holds exactly if i 's local state contains a time 0 node that is labelled with value 0. The rule $K_i \text{not-known}(\exists 0)$ for performing $\text{decide}_i(1)$ holds when i knows that no active process knows $\exists 0$, and we now characterize when this is true. A central role in our analysis will be played by process i 's knowledge about the contents of various nodes in the communication graph. Recall that local states $r_i(m)$ in *fip*'s are communication graphs of the form $\mathcal{G}_\alpha(i, m)$; we abuse notation and write $\theta \in r_i(m)$ (respectively, $(\theta, \theta') \in r_i(m)$) if θ is a node of $\mathcal{G}_\alpha(i, m) = r_i(m)$ (respectively, if (θ, θ') is an edge of $\mathcal{G}_\alpha(i, m) = r_i(m)$); in this case, we say that θ is **seen** by $\langle i, m \rangle$. We now make the following definition:

Definition 3 (Revealed). *Let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$ for a *fip* protocol P . We say that **node $\langle j', m' \rangle$ is revealed to $\langle i, m \rangle$ in r** if either (1) $\langle j', m' \rangle \in r_i(m)$, or (2) for some process i' such that $\langle i', m' \rangle \in r_i(m)$ it is the case that $(\langle j', m' - 1 \rangle, \langle i', m' \rangle) \notin r_i(m)$. We say that **time m' is revealed to $\langle i, m \rangle$ in r** if $\langle j', m' \rangle$ is revealed to $\langle i, m \rangle$ for all processes j' .*

Intuitively, if node $\langle j', m' \rangle$ is revealed to $\langle i, m \rangle$ then i has proof at time m that $\langle j', m' \rangle$ can not carry information that is not known at $\langle i, m \rangle$ but may be known at another node $\langle j, m \rangle$ at the same time. This because either i sees $\langle j', m' \rangle$ at that point—this is part (1)—or i has proof that j' crashed before time m' , and so its state there was \ominus , and j' did not send any messages at or after time m' . It is very simple and straightforward from the definition to determine which nodes are revealed to $\langle i, m \rangle$, based on $r_i(m) = \mathcal{G}_\alpha(i, m)$. Observe that if a node $\langle j', m' \rangle$ is revealed to $\langle i, m \rangle$, then i knows at m what message could have been sent at $\langle j', m' \rangle$: If $\langle j', m' \rangle \in r_i(m)$ then $r_{j'}(m')$ is a subgraph of $r_i(m)$, while if $(\langle j', m' - 1 \rangle, \langle i', m' \rangle) \notin r_i(m)$ for some node $\langle i', m' \rangle \in r_i(m)$, then j' crashed before time m' in r , and so it sends no messages at time m' . Whether and when a node $\langle j', m' \rangle$ is revealed to i depends crucially on the failure pattern. If i receives a message from j' in round $m' + 1$, then $\langle j', m' \rangle$ is immediately revealed to $\langle i, m' + 1 \rangle$. If this message is not received by $\langle i, m' + 1 \rangle$, then $\langle j', m' + 1 \rangle$ — the successor of $\langle j', m' \rangle$ — becomes revealed (as being crashed, i.e. in state \ominus) to $\langle i, m' + 1 \rangle$. But in general $\langle j', m' \rangle$ can be revealed to i at a much later time than $m' + 1$, (A simple instance of this is when $K_i \exists 0$ first becomes true at a time $m > 1$; this happens when $\langle j, 0 \rangle$ with $v_j = 0$ is first revealed to i .)

Suppose that some time $k \leq m$ is revealed to $\langle i, m \rangle$. Then, in a precise sense, process i at time m has all of the information that existed in the system at time k (in the hands of processes that had not crashed by then). In particular, if this information does not contain an initial value of 0, then nobody can know $\exists 0$ at or after time m . We now formalize this intuition and show that revealed nodes can be used to determine when a process can know **not-known**($\exists 0$).

Lemma 6. *Let P be a fip and let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$. For every node $\langle i, m \rangle$, it is the case that $(R_P, r, m) \models K_i \text{not-known}(\exists 0)$ exactly if both (1) $(R_P, r, m) \not\models K_i \exists 0$ and (2) some time $k \leq m$ is revealed to $\langle i, m \rangle$ in r .*

Based on Lemma 6, we now obtain a standard unbeatable consensus protocol for γ_{cr}^t that implements OPT_0 :

Protocol $\text{OPT}_0^{\text{std}}$ (for an undecided process i at time m):

if	i has seen a time-0 node with initial value 0	then decide _i (0)
elseif	some time $k \leq m$ is revealed to $\langle i, m \rangle$	then decide _i (1)

We emphasize that $\text{OPT}_0^{\text{std}}$ (and thus also OPT_0), and all the following protocols, can be implemented efficiently. The protocol only uses information about the existence of 0 and about the rounds at which processes crash. It can therefore be implemented in such a way that any process sends a total of $O(f \log n)$ bits (see Lemma 23 in Appendix A.5) in every run, and executes $O(n)$ local steps in every round.

The formulation of $\text{OPT}_0^{\text{std}}$, in addition to facilitating an efficient implementation, also makes the worst-case stopping time of $\text{OPT}_0^{\text{std}}$ and OPT_0 apparent.

Lemma 7. *In $\text{OPT}_0^{\text{std}}$ (and thus also OPT_0), all decisions are made by time $f + 1$ at the latest.²*

It is interesting to compare OPT_0 with efficient early-stopping consensus protocols [2, 5, 11, 14]. Let's say that **the sender set repeats at** $\langle i, m \rangle$ in run r if i hears from the same set of processes in rounds $m - 1$ and m . If this happens then, for every $\langle j, m - 1 \rangle \notin r_i(m)$, we are guaranteed that $(\langle j, m - 2 \rangle, \langle i, m - 1 \rangle) \notin r_i(m)$. Thus, all nodes at time $(m - 1)$ are revealed to $\langle i, m \rangle$. Indeed, in a run in which f failures actually occur, the sender set will repeat for every correct process by time $f + 1$ at the latest. Efficient early stopping protocols typically decide when the sender set

²In all our protocols, a process can stop at the earlier of one round after deciding and time $t + 1$.

repeats. Indeed, the protocol $P0_{\text{opt}}$ that was claimed by [14] to be unbeatable does so as well, with a slight optimization. Writing $\forall 1$ to stand for “all initial values are 1”, $P0_{\text{opt}}$ is described as follows:

Protocol $P0_{\text{opt}}$ (for an undecided process i at time m) [14] :

if $K_i \exists 0$ **then** $\text{decide}_i(0)$
elseif $K_i \forall 1$ or $m \geq 2$ and the sender set repeats at $\langle i, m \rangle$ **then** $\text{decide}_i(1)$

OPT_0 and $P0_{\text{opt}}$ differ only in the rule for deciding 1. But OPT_0 strictly beats $P0_{\text{opt}}$, and sometimes by a wide margin. If $t = \Omega(n)$ then it can decide faster by a *ratio* of $\Omega(n)$. Indeed, we can show:

Lemma 8. *If $3 \leq t \leq n - 2$, then OPT_0 strictly dominates $P0_{\text{opt}}$. Moreover, there exists an adversary for which $\text{decide}_i(1)$ is performed after 3 rounds in OPT_0 , and after $t + 1$ rounds in $P0_{\text{opt}}$.*

3.2 Hidden Paths and Agreement

It is instructive to examine the proof of Lemma 6 (see Appendix A.1) and consider when an active process i is undecided at $\langle i, m \rangle$ in OPT_0 . This occurs if both $\neg K_i \exists 0$ and, in addition, for every $k = 0, \dots, m$ there is at least one node $\langle j_k, k \rangle$ that is not revealed to $\langle i, m \rangle$. We call the sequence of nodes $\langle j_0, 0 \rangle, \dots, \langle j_m, m \rangle$ a **hidden path w.r.t. $\langle i, m \rangle$** . Such a hidden path implies that all processes j_0, \dots, j_m have crashed. Roughly speaking, $\exists 0$ could be relayed along such a hidden path without i knowing it (see Fig. 1). More formally, its existence means that there is a run,

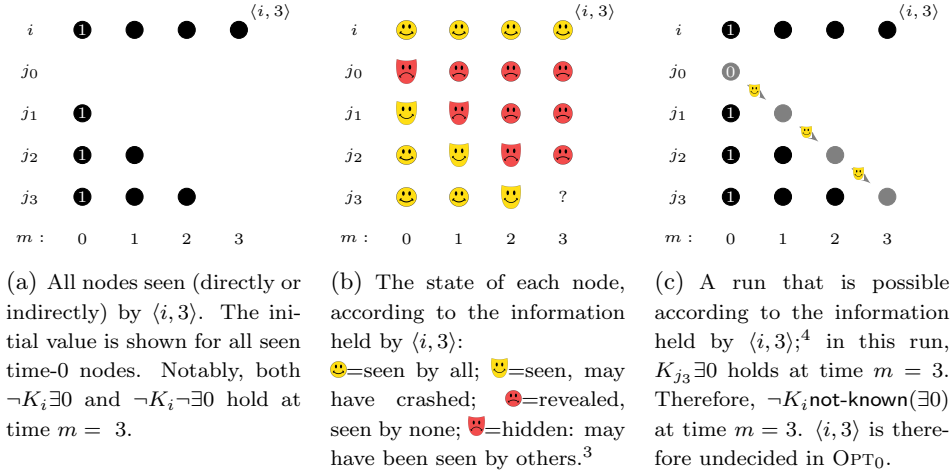


Figure 1: A hidden path $\langle j_0, 0 \rangle, \dots, \langle j_3, 3 \rangle$ w.r.t. $\langle i, 3 \rangle$ implies $\neg K_i \text{not-known}(\exists 0)$ at 3.

indistinguishable at $\langle i, m \rangle$ from the current one, in which $v_{j_0} = 0$ and this fact is sent from each j_k to j_{k+1} in every round $k + 1 \leq m$. In that run process j_m is active at time m and $K_{j_m} \exists 0$, and that is why $K_i \text{not-known}(\exists 0)$ *does not* hold. Hidden paths are implicit in many lower bound proofs

³For simplicity, in this example every node seen by $\langle i, 3 \rangle$ is also seen by all other nodes in the view of $\langle i, 3 \rangle$. In other words, there exists no node $\langle j, m' \rangle$ that is in state 👁 according to the information held by $\langle i, 3 \rangle$, i.e. both $\langle j, m' \rangle$ is seen by $\langle i, 3 \rangle$, and i has indirectly learnt by time 3 that j has in fact crashed at m' .

⁴In this run, the state of both $\langle j_0, 0 \rangle$ and $\langle j_1, 1 \rangle$, according to the information held by $\langle j_3, 3 \rangle$, is 👁 , as defined in Footnote 3.

for consensus in the crash failure model [5, 8], but they have never before been captured formally. Clearly, hidden paths can relay more than just the existence of a value of 0. In a protocol in which some view can prove that the state is univalent in the sense of Fischer, Lynch and Paterson [10], a hidden path from a potentially pivotal state can keep processes from deciding on the complement value. Our analysis in the remainder of the paper provides additional cases in which unbeatable consensus is obtained when hidden paths can be ruled out.

3.3 Majority Consensus

Can we obtain other unbeatable consensus protocols? Clearly, the symmetric protocol OPT_1 , obtained from OPT_0 by reversing the roles of 0 and 1, is unbeatable and neither dominates, nor is dominated by, OPT_0 . Of course, OPT_0 and OPT_1 are extremely biased, each deciding on its favourite value if at all possible, even if it appears as the initial value of a single process. One may argue that it is natural, and may be preferable in many applications, to seek a more balanced solution, in which minority values are not favoured. Fix $n > 0$ and define the fact “ $\text{Maj} = 0$ ” to be true if at least $n/2$ initial values are 0, while “ $\text{Maj} = 1$ ” is true if strictly more than $n/2$ values are 1. Finally, relative to a node $\langle i, m \rangle$, we define $\text{Maj}\langle i, m \rangle \triangleq 0$ if at least half of the processes whose initial value is known to i at time m have initial value 0; $\text{Maj}\langle i, m \rangle \triangleq 1$ otherwise. Consider the following protocol:

Protocol OPT_{Maj} (for an undecided process i at time m):

if	$K_i(\text{Maj} = 0)$	then	$\text{decide}_i(0)$
elseif	$K_i(\text{Maj} = 1)$	then	$\text{decide}_i(1)$
elseif	some time $k \leq m$ is revealed to $\langle i, m \rangle$	then	$\text{decide}_i(\text{Maj}\langle i, m \rangle)$.

We note that whether $K_i(\text{Maj} = 0)$ (resp. $K_i(\text{Maj} = 1)$) holds can be checked efficiently: it holds exactly if i has seen at least (resp. strictly more than) $n/2$ time-0 nodes with initial value 0 (resp. 1).

Theorem 3. *If $t > 0$, then OPT_{Maj} is an unbeatable consensus protocol. In particular, in a run in which $f \leq t$ failures actually occur, all decisions are performed by time $f + 1$, at the latest.*

The proof of Theorem 3 formalizes the following idea. Suppose that i sees fewer than a full majority of either value at $\langle i, m \rangle$ and has a hidden path. Then i considers it possible that the node $\langle j_1, 1 \rangle$ in the hidden path may have seen either a full majority of 0’s or a full majority of 1’s, and this information may reach an active node $\langle j_m, m \rangle$. Decision is thus impossible in this case, and decisions are made when no hidden path w.r.t. $\langle i, m \rangle$ is possible. Thus, OPT_{Maj} is an unbeatable consensus protocol that satisfies an additional “fairness” property:

Majority Validity: For $v \in \{0, 1\}$, if more than half of the processes are both correct and have initial value v , then no process decides \bar{v} in r .

4 Unbeatable Uniform Consensus

It is often of interest to consider *uniform* consensus [2, 7, 12, 16, 20, 21] in which we replace the **Agreement** condition of consensus by:

Uniform Agreement: The processes that decide in a given run must all decide on the same value.

This forces correct processes and faulty ones to act in a consistent manner. Requiring uniformity makes sense only in a setting where failures are benign, and all processes that decide do so according to the protocol. Uniformity may be desirable when elements outside the system can observe decisions, as in distributed databases when decisions correspond to commitments to values.

Under crash failures, a process generally does not know whether or not it is correct. Indeed, so long as it has not seen t failures, the process may (for all it knows) crash in the future. As a result, while $K_i \exists 0$ is a necessary condition for $\text{decide}_i(0)$ as before, it cannot be a *sufficient* condition for decision in any uniform consensus protocol. This is because a process starting with 0 immediately decides 0 with this rule, and may immediately crash. If all other processes have initial value 1, all other decisions can only be on 1. Of course, $K_i \exists 0$ is still a necessary condition for deciding 0, but it is *not* sufficient. Denote by $\exists \text{correct}(v)$ the fact “some **correct** process knows $\exists v$ ”. We show the following:

Lemma 9. $K_i \exists \text{correct}(v)$ is a necessary condition for i deciding v in any protocol solving Uniform Consensus.

There is a direct way to test whether $K_i \exists \text{correct}(v)$ holds, based on $r_i(m)$:

Lemma 10. Let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$ and assume that i knows of d failures at (r, m) . Then $(R_P, r, m) \models K_i \exists \text{correct}(v)$ iff at least one of (a) $m > 0$ and $(R_P, r, m-1) \models K_i \exists v$, or (b) $(R_P, r, m) \models K_i(K_j \exists v \text{ held at time } m-1)$ holds for at least $(t-d)$ distinct processes j .

By Lemma 3, at time $t+1$ the conditions $K_i \exists v$ and $K_i \exists \text{correct}(v)$ are equivalent. As in the case of consensus, we note that if $K_i \exists 0$ (equivalently, $K_i \exists \text{correct}(0)$) does not hold at time $t+1$, then it never will. We thus phrase the following *beatable* algorithm, analogous to P_0 , for Uniform Consensus; in this protocol, $K_i \exists \text{correct}(0)$ (the necessary condition for deciding 0 in uniform consensus) replaces $K_i \exists 0$ (the necessary condition in consensus) as the decision rule for 0. The decision rule for 1 remains the same. Note that $K_i \exists \text{correct}(0)$ can be efficiently checked, by applying the test of Lemma 10.

Protocol U- P_0 (for an undecided process i at time m):

if $K_i \exists \text{correct}(0)$ **then** $\text{decide}_i(0)$
elseif $m = t+1$ **then** $\text{decide}_i(1)$.

Following a similar line of reasoning to that leading to OPT_0 , we obtain an unbeatable uniform consensus protocol:

Protocol U- OPT_0 (for an undecided process i at time m):

if $K_i \exists \text{correct}(0)$ **then** $\text{decide}_i(0)$
elseif $\neg K_i \exists 0$ and some time $k \leq m$ is revealed to $\langle i, m \rangle$ **then** $\text{decide}_i(1)$.

Recall that whether $K_i \exists \text{correct}(0)$ holds can be checked efficiently via the characterization in Lemma 10.

Theorem 4. U- OPT_0 is an unbeatable **uniform** consensus protocol in which all decisions are made by time $f+2$ at the latest, and if $f \geq t-1$, then all decisions are made by time $f+1$ at the latest.

Hidden paths again play a central role. Indeed, as in the construction of OPT_0 from P_0 , the construction of U- OPT_0 from U- P_0 involves some decisions on 1 being moved earlier in time, by means of the last condition, checking the absence of a hidden path. (Decisions on 0 cannot be moved any earlier, as they are taken as soon as the necessary condition for deciding 0 holds.) Observe that the need to obtain $K_i \exists \text{correct}(v)$ rather than $K_i \exists v$ concisely captures the essential

distinction between uniform consensus and nonuniform consensus. The fact that the same condition — the existence of a hidden path — keeps a process i from knowing that no active j can know $K_j \exists \text{correct}(v)$, as well as keeping i from knowing that no j knows $K_j \exists v$, explains why the bounds for both problems, and their typical solutions, are similar.

Proving the unbeatability of U-OPT_0 is more challenging than proving it for OPT_0 . Intuitively, this is because gaining that an initial value of 0 that is known by a nonfaulty process does not imply that some process has already decided on 0. As a result, the possibility of dominating U-OPT_0 by switching 0 decisions to 1 decisions needs to be explicitly rejected. This is done by employing reachability arguments essentially establishing the existence of the continual common knowledge conditions of [14].

The fastest early-stopping protocol for uniform consensus in the literature, opt-EDAUC of [2] (a similar algorithm is in [7]), also stops in $\min(f+2, t+1)$ rounds at the latest. Similarly to Lemma 8, not only does U-OPT_0 strictly dominate opt-EDAUC , but furthermore, there are adversaries against which U-OPT_0 decides in 1 round, while opt-EDAUC decides in $t+1$ rounds:

Lemma 11. *If $2 \leq t \leq n-2$, then U-OPT_0 strictly dominates the opt-EDAUC protocol of [2]. Moreover, there exists an adversary for which $\text{decide}_i(1)$ is performed after 1 round in U-OPT_0 , and after $t+1$ rounds in opt-EDAUC .*

5 Discussion

It is possible to consider variations on the notion of unbeatability. One could, for example, compare runs in terms of the time at which the last correct process decides. We call the corresponding notion ***last-decider unbeatability***.⁵ This neither implies, nor is implied by, the notion of unbeatability studied so far in this paper. None of the consensus protocols in the literature is last-decider unbeatable. In fact, all of our protocols are also last-decider unbeatable:

Theorem 5. *The protocols OPT_0 and OPT_{Maj} are also last-decider unbeatable for consensus, while U-OPT_0 is last-decider unbeatable for uniform consensus.*

We note that Lemmas 8 and 11 show that our protocols beat the previously-known best ones by a large margin w.r.t. last-decider unbeatability as well.

Unbeatability is a natural optimality criterion for distributed protocols. It formalizes the intuition that a given protocol cannot be strictly improved upon, which is significantly stronger than saying that it is worst-case optimal, or even early stopping. All of the protocols that we have presented have a very concise and intuitive description, and are efficiently implementable; thus, unbeatability is attainable at a modest price. Crucially, our unbeatable protocols can decide much faster than previously known solutions to the same problems.

Acknowledgements

Armando Castañeda was supported in part by an Aly Kaufman Fellowship at the Technion. Yannai Gonczarowski was supported in part by ISF grant 230/10, by the Google Inter-university center for Electronic Markets and Auctions, by the European Research Council under the European Community’s Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. [249159] and by an Adams Fellowship of the Israeli Academy of Sciences and Humanities. Yoram Moses is the Israel Pollak Academic chair at the Technion; his work was supported in part by ISF grant 1520/11.

⁵This notion was suggested to us by Michael Schapira; we thank him for the insight.

References

- [1] A. Castañeda, Y. A. Gonczarowski, and Y. Moses. Brief announcement: Pareto-optimal solutions to consensus and set consensus. In *PODC*, pages 113–115, 2013.
- [2] B. Charron-Bost and A. Schiper. Uniform consensus is harder than consensus. *J. Algorithms*, 51(1):15–37, 2004.
- [3] B. Coan. A communication-efficient canonical form for fault-tolerant distributed protocols. In *Proc. 5th ACM Symp. on Principles of Distributed Computing*, pages 63–72, 1986.
- [4] D. Dolev. Beep protocols (personal communication).
- [5] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in Byzantine agreement. *Journal of the ACM*, 34(7):720–741, 1990.
- [6] D. Dolev and H. R. Strong. Requirements for agreement in a distributed system. In H. J. Schneider, editor, *Distributed Data Bases*, pages 115–129. North-Holland, 1982.
- [7] P. Dutta, R. Guerraoui, and B. Pochon. The time-complexity of local decision in distributed agreement. *SIAM J. Comput.*, 37(3):722–756, 2007.
- [8] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.
- [9] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 2003.
- [10] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty processor. *Journal of the ACM*, 32(2):374–382, 1985.
- [11] E. Gafni, R. Guerraoui, and B. Pochon. The complexity of early deciding set agreement. *SIAM J. Comput.*, 40(1):63–78, 2011.
- [12] V. Hadzilacos. On the relationship between the atomic commitment and consensus problems. In *Fault-Tolerant Distributed Computing*, pages 201–208, 1986.
- [13] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3):549–587, 1990. A preliminary version appeared in *PODC*, 1984.
- [14] J. Y. Halpern, Y. Moses, and O. Waarts. A characterization of eventual byzantine agreement. *SIAM J. Comput.*, 31(3):838–865, 2001.
- [15] M. Herlihy, Y. Moses, and M. R. Tuttle. Transforming worst-case optimal solutions for simultaneous tasks into all-case optimal solutions. In *PODC*, pages 231–238, 2011.
- [16] I. Keidar and S. Rajsbaum. A simple proof of the uniform consensus synchronous lower bound. *Inf. Process. Lett.*, 85(1):47–52, 2003.
- [17] Y. Moses. *Knowledge and Distributed Coordination*. in preparation.
- [18] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

- [19] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.
- [20] M. Raynal. Optimal early stopping uniform consensus in synchronous systems with process omission failures. In *SPAA*, pages 302–310. ACM Press, 2004.
- [21] X. Wang, Y. M. Teo, and J. Cao. A bivalency proof of the lower bound for uniform consensus. *Inf. Process. Lett.*, 96(5):167–174, 2005.

A Proofs

A.1 Consensus

Proof of Lemma 1. This proof uses notation introduced in Section 2. Let P be a consensus protocol and let $R_P = R(P, \gamma_{\text{cr}}^t)$. Let $v \in V$, let $r \in R_P$ and let $\langle i, m \rangle$ be a node s.t. i decides on v at time m in r .

We commence by proving (a). Assume for contradiction that no process has initial value v in r . By definition of γ_{cr}^t , there exists a run r' of P , s.t. 1) $r'_i(m) = r_i(m)$, 2) i does not fail in r' , and 3) The initial values in r' are the same as in r . As $r'_i(m) = r_i(m)$, we have that i decides on v at time m in r' as well. As the initial values in r' are the same as in r , we have that no process has initial value v in r' . As i does not fail in r' , we therefore have that **Validity** does not hold regarding the decision of i in r' — a contradiction.

We move on to proving (b). Assume for contradiction that some process j decides \bar{v} at some time $m' \leq m$ in r , and that j is active at m in r . Once again by definition of γ_{cr}^t , there exists a run r' of P , s.t. 1) $r'_i(m) = r_i(m)$, 2) $r'_j(m') = r_j(m')$, and 3) neither i nor j fail in r' . As $r'_i(m) = r_i(m)$, we have that i decides on v at time m in r' as well; as $r'_j(m') = r_j(m')$, we have that j decide on \bar{v} at time m' in r' as well. As neither i nor j fail in r' , we therefore have that **Agreement** does not hold in r' — a contradiction. \square

Proof of Lemma 2. Directly from Lemma 1 and Theorem 1. \square

While Lemma 3 is given and proved in [8], for completeness we reprove it here using the notation and machinery of this paper; this proof is assisted by Definition 4 and Lemma 12.

Definition 4. Let P be a protocol in γ_{cr}^t and let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$. Let $v \in V$ and let $\langle i, m \rangle$ be a node. We say that **there is a v -chain for $\langle i, m \rangle$ in the run r** if, for some $d \leq m$, there is a sequence $j_0, j_1, \dots, j_d = i$ of distinct processes, such that $v_{j_0} = v$ and for all $1 \leq k \leq d$, the process j_k receives a message from j_{k-1} at time k in r .

Lemma 12. Let P be a fip in γ_{cr}^t and let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$. Then for every processes i and time $m \geq 0$, it is the case that $(R_P, r, m) \models K_i \exists 0$ iff there is a 0-chain for $\langle i, m \rangle$ in r .

Proof. For the first direction, assume that there is a 0-chain $j_0, \dots, j_d = i$ for $\langle i, m \rangle$ in r . It is easy to show by induction that $K_{j_k} \exists 0$ at k in r for every k ; therefore, $K_i \exists 0$ at d in r , and since P is a fip, $K_i \exists 0$ at m in r , as required. We prove the second direction for all i by induction on m .

Base ($m = 0$): Since process i at time 0 knows no initial value but its own, we have that $v_i = 0$ and so i (with $d = 0$) is a 0-chain as required.

Inductive step ($m > 0$): In a fip, $K_i \exists 0$ at m implies that either $K_i \exists 0$ at $m - 1$ or $K_j \exists 0$ at $m - 1$ for some $j \neq i$ that successfully sends a message at time $m - 1$ to i . If $K_i \exists 0$ at $m - 1$, then by the induction hypothesis there exists a 0-chain for $\langle i, m - 1 \rangle$ in r , and by definition this is also

a 0-chain for $\langle i, m \rangle$ in r . It remains to consider the case in which $K_i \exists 0$ does not hold at $m - 1$; therefore, $K_j \exists 0$ at $m - 1$ for some j that successfully sends a message at time $m - 1$ to i . By the induction hypothesis, there exists a 0-chain $j_0, \dots, j_d = j$ for $\langle j, m - 1 \rangle$. We first claim that i does not appear in that chain; indeed, if $j_{d'} = i$ for some $d' < d$, then by definition $j_0, \dots, j_{d'}$ would be a 0-chain for $\langle i, m - 1 \rangle$, and by the previous direction we would have $K_i \exists 0$ at $m - 1$ in r . We now claim that $d = m - 1$; indeed, if $d < m - 1$, then j_0, \dots, j_d would be a 0-chain for $\langle j, d \rangle$, and so we would have $K_j \exists 0$ at $d < m - 1$. As j is active at all times earlier than $m - 1$, we would have that $\langle j, d \rangle$ successfully sends a message to i , and so $K_i \exists 0$ at $d + 1 \leq m - 1$; as P is a fip, we would therefore have that $K_i \exists 0$ at $m - 1$ — a contradiction. As i does not appear in j_0, \dots, j_d , and as $d = m - 1$, by definition j_0, \dots, j_d, i is a 0-chain for i , as required. \square

Proof of Lemma 3. Assume that $(R_P, r, t + 1) \models K_i \exists v$. By Lemma 12, there exists a 0-chain j_0, \dots, j_d for $\langle i, t + 1 \rangle$. If j appears in j_0, \dots, j_d , then by Lemma 12 we are done; assume, therefore, that j does not appear in j_0, \dots, j_d . If $d < t + 1$, then since i successfully sends all messages at times earlier than $t + 1$, we have that j_0, \dots, j_d, j is a 0-chain for $\langle j, t + 1 \rangle$; therefore, by Lemma 12, $K_j \exists v$ at $t + 1$, as required. Otherwise, $d = t + 1$, and so, as j_0, \dots, j_{d-1} are $t + 1$ distinct processes, there exists $0 \leq d' \leq d - 1$ s.t. $j_{d'}$ is nonfaulty throughout r . Therefore, $j_0, \dots, j_{d'}, j$ is a 0-chain for $\langle j, t + 1 \rangle$, as required. \square

Proof of Lemma 4. Assume that $Q \preceq P_0$ solves consensus; w.l.o.g., Q is a fip as well. We prove the claim for all processes i and adversaries α , by induction on the time m at which $K_i \exists 0$ first holds in $Q[\alpha]$ (and, equally, in $P_0[\alpha]$).

Base ($m = 0$): As i decides 0 at time 0 in $P_0[\alpha]$, by Lemma 2 we have $K_i \exists 0$ at time 0 in $P_0[\alpha]$ (and so also in $Q[\alpha]$). Since process i at time 0 knows no initial value but its own, it follows that i is assigned an initial value of 0 by α . Hence, $K_i \exists 1$ does *not* hold at 0. By Lemma 2, i therefore does not decide 1 at time 0 in $Q[\alpha]$. Since i decides at time 0 in $P_0[\alpha]$, it must decide at time 0 in $Q[\alpha]$ as well, and so decides 0, as required.

Inductive step ($m > 0$): Assume that the claim holds for all times $< m$. Recall that m is the first time at which $K_i \exists 0$ holds. In a fip, this can only happen if $K_i \exists 0$ does not hold at time $m' < m$ and i receives at time m a message with a 0 from some process j that is active at time $m - 1$. Thus, $K_j \exists 0$ holds at time $m - 1$, and by the induction hypothesis, j decides 0 when $K_j \exists 0$ first holds in $Q[\alpha]$ — denote this time by m' ; as $K_j \exists 0$ holds at time $m - 1$, we have $m' \leq m - 1$. Observe that in γ_{cr}^t , if i receives a message from j in round m , then i cannot know that j is faulty at time m ; more precisely, denoting by β the adversary that never crashes i nor j at all, and that otherwise agrees with α (this is a legal adversary, as it specifies no more than t crash failures), we have in the run $r' = Q[\beta]$ that 1) $r'_i(m) = r_i(m)$, 2) $r'_j(m') = r_j(m')$, and 3) neither i nor j fail. Since Q satisfies **Agreement**, i cannot decide 1 during $Q[\beta]$, and therefore cannot decide 1 at or before time m during $Q[\alpha]$. Moreover, by Lemma 2, $K_i \exists 0$ is a precondition for process i deciding 0, and so i cannot decide 0 before time m during $Q[\alpha]$. Since Q dominates P_0 , we have that i must decide by time m in $Q[\alpha]$, and therefore it decides 0 at m in $Q[\alpha]$. \square

Proof of Lemma 5. \implies : Assume that $(R_P, r, m) \not\models K_i \text{not-known}(\exists 0)$. Therefore, by definition of K_i , there exists a run $r' \in R_P$ s.t. 1) $r'_i(m) = r_i(m)$, and 2) $(R_P, r', m) \not\models \text{not-known}(\exists 0)$. As $(R_P, r', m) \not\models \text{not-known}(\exists 0)$, there exists a process j s.t. $K_j \exists 0$ holds at m in r' (and j is active at m in r'). By definition, $K_j \exists 0$ first holds at or before time m in r' , and so j decides 0 before or at time m in r' ; therefore, $(R_P, r', m) \not\models \text{no-decided}(0)$. As $r'_i(m) = r_i(m)$, we therefore have $(R_P, r, m) \not\models K_i \text{no-decided}(0)$, as required.

\impliedby : We will show that $(R_P, r, m) \models \text{not-known}(\exists 0)$ implies $(R_P, r, m) \models \text{no-decided}(0)$; by definition of knowledge, it will then follow that $(R_P, r, m) \models K_i \text{not-known}(\exists 0)$ implies $(R_P, r, m) \models$

$K_i \text{no-decided}(0)$. Assume, therefore, that $(R_P, r, m) \models \text{not-known}(\exists 0)$, and let j be a process that is active at time m in r . As $\text{not-known}(\exists 0)$ at m in r , we have that $K_j \exists 0$ does not hold at m in r . As P is a fip, we have that neither does $K_j \exists 0$ hold at any time prior to m in r . By definition, therefore j does not decide 0 before or at m in r , as required. \square

Lemma 13. *Let P be a fip in γ_{cr}^t and let $r \in R_P = R(P, \gamma_{\text{cr}}^t)$. Let i be a process. If $(R_P, r, t+1) \not\models K_i \exists 0$, then $(R_P, r, t+1) \models K_i \text{not-known}(\exists 0)$.*

Proof. By Lemma 3, we have that $\neg K_i \exists 0$ at time $t+1$ implies $\text{not-known}(\exists 0)$ at that time; by definition of knowledge, we therefore have that $K_i(\neg K_i \exists 0 \wedge m = t+1)$ implies $K_i \text{not-known}(\exists 0)$. As both the clock m and the value of t are common knowledge, we therefore have that $K_i(\neg K_i \exists 0)$ at time $t+1$ implies $K_i \text{not-known}(\exists 0)$ at that time. Finally, by the definition of knowledge we have that $K_i(\neg K_i \exists 0)$ holds iff $\neg K_i \exists 0$ holds, and the proof is complete. \square

Theorem 6. OPT_0 solves consensus in γ_{cr}^t .

Proof. In some run r of OPT_0 , let i be a nonfaulty process.

Decision: By definition of OPT_0 , for any process that is active at time $t+1$, if i has not decided 0 by that time, we have $\neg K_i \exists 0$ at that time. Therefore, by Lemma 13, we have that $K_i \text{not-known}(\exists 0)$ at that time and so i decides upon 1 if it is undecided. Therefore, all processes that are active at time $t+1$, and in particular all nonfaulty processes, decide by that time at the latest, and in particular decide at some point throughout the run, as required.

Henceforth, let m be the decision time of i and let v be the value upon which i decides.

Validity: If $v = 0$, then $K_i \exists 0$ at m ; thus, $\exists 0$ as required. Otherwise, $K_i \exists 0$ does not hold at m ; therefore, the initial value of i is 1, and so $\exists 1$ as required.

Agreement: It is enough to show that if $v = 1$, then no correct process ever decides 0 in the current run. Indeed, if any nonfaulty process j decided 0 at some time $m' < m$, then i would have received a message with a 0 from j at $m' + 1 \leq m$, and so we would have $K_i \exists 0$ at m . To complete the proof, it is enough to show that no process decides 0 at any time $m' \geq m$; this follows by an easy inductive argument, using the fact that $\text{not-known}(\exists 0)$ at any time m'' implies $\text{not-known}(\exists 0)$ at $m'' + 1$. \square

Proof of Theorem 2. Correctness is shown in Theorem 6. We thus have to show that for every protocol consensus protocol $Q \preceq \text{OPT}_0$, we also have $\text{OPT}_0 \preceq Q$. Let, therefore, Q be a consensus protocol s.t. $Q \preceq \text{OPT}_0$; w.l.o.g., Q is a fip.

We first claim that $\text{OPT}_0 \preceq P_0$. Indeed, whenever P_0 decides upon 0, so does OPT_0 ; let therefore i be a process deciding upon 1 in P_0 ; by definition of P_0 , this decision is made at time $m = t+1$, and furthermore, $\neg K_i \exists 0$ at that time. By Lemma 13, we therefore have that $K_i \text{not-known}(\exists 0)$ at time m , and so, i decides upon 1 in OPT_0 at that time if it has not already decided.

By transitivity of domination, we thus have that $Q \preceq P_0$. By Lemma 4, we therefore have that $\text{decide}_i(0)$ is performed in Q exactly when $K_i \exists 0$ first holds; therefore, no decision on 0 is made in Q before OPT_0 . Moreover, by Lemmas 2 and 5, we therefore have that $K_i \text{not-known}(\exists 0)$ is a necessary conditions for $\text{decide}_i(1)$ in $R_Q = R(Q, \gamma_{\text{cr}}^t)$. Therefore, no decision on 1 is made in Q before OPT_0 . Therefore $\text{OPT}_0 \preceq Q$, as required, and the proof is complete. \square

Proof of Lemma 6. We first claim that $(R_P, r, m) \models \text{not-known}(\exists 0)$ iff for every $0 \leq k \leq m$, there exists a process j_k s.t. $K_{j_k} \exists 0$ at time k in r — we call such j_0, \dots, j_m a **0-path** for time m in r ; the proof is similar to (and simpler than) that of Lemma 12 and is left to the reader.

Assume first that some time $k \leq m$ is revealed to $\langle i, m \rangle$ in r . As $(R_P, r, m) \not\models K_i \exists 0$, we thus have that no time- k node j satisfies $K_j \exists 0$; therefore, no 0-path exists for time m in r , and so

$(R_P, r, m) \models \text{not-known}(\exists 0)$. Note that by definition of knowledge, time k is revealed to $\langle i, m \rangle$ in r iff $(R_P, r, m) \models K_i(\text{time } k \text{ is revealed to } \langle i, m \rangle \text{ in the current run})$. Therefore, we have that time k being revealed to $\langle i, m \rangle$ implies not only $(R_P, r, m) \models \text{not-known}(\exists 0)$, but also $(R_P, r, m) \models K_i \text{not-known}(\exists 0)$, as required.

Assume now that no time $k \leq m$ is revealed to $\langle i, m \rangle$, i.e. that for every $k \leq m$, there exists a time- k node $\langle j_k, k \rangle$ that is not revealed to $\langle i, m \rangle$ in r — in Section 3.2, we call such j_0, \dots, j_m a **hidden path** w.r.t. $\langle i, m \rangle$ in r . We construct a run $r' \in R_P$ s.t. $r'_i(m) = r_i(m)$, in which j_0, \dots, j_m constitutes a 0-path for time m — see Fig. 1 in Section 3.2. The adversary in r' meets the following conditions, and otherwise coincides with that of r :

- The initial value of j_0 is 0.
- For every $k < m$, the node $\langle j_k, k \rangle$ crashes, successfully sending a message solely to $\langle j_{k+1}, k+1 \rangle$.
- $\langle j_m, m \rangle$ is nonfaulty.

It is straightforward to verify that $r'_i(m) = r_i(m)$, that no more crashes occur in r' than in r , and that j_0, \dots, j_m indeed is a 0-path for time m in r . As $(R_P, r', m) \not\models \text{not-known}(\exists 0)$, and as $r'_i(m) = r_i(m)$, we therefore have that $(R_P, r, m) \not\models K_i \text{not-known}(\exists 0)$, as required. \square

Proof of Lemma 7. Let i be an undecided node at time m in $\text{OPT}_0^{\text{std}}$; it is enough to show that $m \leq f$. As i is undecided, by definition of $\text{OPT}_0^{\text{std}}$, for every $0 \leq k < m$, there exists a node process j_k s.t. $\langle j_k, k \rangle$ is not revealed to $\langle i, m \rangle$. We first note that all of the nodes j_k are faulty; indeed, as $\langle j_k, k \rangle$ is not revealed to $\langle i, m \rangle$, and as $k < m$, we have that $\langle i, k+1 \rangle$ receives no message from $\langle j_k, k \rangle$. We further note that all j_k are distinct; indeed, for every $k < k' < m$, we have (once again since $\langle j_k, k \rangle$ is not revealed to $\langle i, m \rangle$) that $(\langle j_k, k'-1 \rangle, \langle i, k' \rangle) \notin r_i(m)$ while $\langle i, k' \rangle \in r_i(m)$, and so by definition $\langle j_k, k' \rangle$ is revealed to $\langle i, m \rangle$. We conclude that j_0, \dots, j_{m-1} are m distinct faulty nodes, and so $m \leq f$ and the proof is complete. \square

Proof of Lemma 8. First notice that OPT_0 dominates $P0_{\text{opt}}$, since $K_i \forall 1$ is true iff time 0 is revealed to i , and if i 's sender set repeats in round m , then time $m-1$ is revealed to i at time m . Hence, for every adversary, processes decide in OPT_0 at least as soon as they do in $P0_{\text{opt}}$. We now show an adversary for which the decisions are made strictly earlier in OPT_0 than in $P0_{\text{opt}}$; moreover, this adversary meets the conditions of the second clause of the lemma.

Denote the processes by $\text{Procs} = \{1, 2, \dots, n\}$. Let α be defined as follows. All initial values in α are 1. In round 1, only process 1 fails, and it is silent: it crashes without sending any messages. In round 2 two processes crash—process 2 and process 3, with process 2 sending only to process n , and process 3 sending to everyone except process n . No process fails in round 3, and, in each of the rounds $m = 4, \dots, t$, process m crashes without sending any messages. Since precisely t processes fail in α we have that $\alpha \in \text{Crash}(t)$.

Observe that in $\text{fip}[\alpha]$ no correct process ever knows process 1's initial value. In addition, for every correct process, the first round in which the sender set repeats is round $t+1$. Indeed, every correct process other than n fails to hear from process m for the first time in round m , for $m = 1, \dots, t$, while process n differs slightly, in that it fails to hear from process 3 in round 2 and from process 2 in round 3. Therefore, in $P0_{\text{opt}}[\alpha]$ all correct processes decide 1 at time $t+1$, since round $t+1$ is the first one in which their sender set repeats; no process decides any earlier. Now let us consider when a process i that is correct according to α decides in OPT_0 . By definition, i receives messages in round 3 from both $\langle n, 2 \rangle$ and $\langle n-1, 2 \rangle$. Together, these contain the information about nodes $\langle 2, 1 \rangle, \langle 3, 1 \rangle, \dots, \langle n, 1 \rangle$. Moreover, node $\langle 1, 1 \rangle$ is revealed to $\langle i, 3 \rangle$ as well (as being crashed — \odot), since the edge $(\langle 1, 0 \rangle, \langle i, 1 \rangle)$ is absent from i 's view at $\langle i, 3 \rangle$. It follows that time 1 is revealed

to $\langle i, 3 \rangle$, and so i decides 1 at time 3, after 3 rounds, as claimed. Since $3 < 4 \leq t + 1$, we have that when the adversary is α , decisions in OPT_0 occur strictly earlier than in $P0_{\text{opt}}$, and we are done. \square

A.2 Majority Consensus

The proof of Theorem 3 is based on two lemmas:

Lemma 14 (Decision at time 1). *Assume that $n > 2$ and $t > 0$. Let $Q \preceq \text{OPT}_{\text{Maj}}$ solve Consensus and let $r = Q[\alpha]$ be a run of Q . Let i be a process and let v be a value. If $K_i(\text{Maj} = v)$ at $(r, 1)$, then Q makes i decide v before or at time 1 in r .*

Proof. By definition of OPT_{Maj} , i decides in $\text{OPT}_{\text{Maj}}[\alpha]$ by time 1, since $K_i(\text{Maj} = v)$ holds at $(\text{OPT}_{\text{Maj}}[\alpha], 1)$. As $Q \preceq \text{OPT}_{\text{Maj}}$, we thus have that i must decide upon some value in $r = Q[\alpha]$ before or at time 1. Thus, it is enough to show that i cannot decide $1-v$ up to time 1 in r .

We prove the claim by induction on $n - |Z_i|$, where Z_i is defined to be the set of processes k with initial value v , s.t. $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$. As $K_i(\text{Maj} = v)$ at $(r, 1)$, we have $|Z_i| \geq \frac{n}{2}$ and so $2 \leq |Z_i| \leq n$.

Base: $|Z_i| = n$. In this case, all initial values are v , and so by **Validity** i cannot decide $1-v$ in r .

Step: Let $2 \leq \ell < n$ and assume that the claim holds whenever $|Z_i| = \ell + 1$. Assume that $|Z_i| = \ell$. As $|Z_i| \geq 2$, there exists $j \in Z_i \setminus \{i\}$. We reason by cases.

- I. If there exists a process k s.t. $\langle k, 0 \rangle$ is hidden from $\langle i, 1 \rangle$, then there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) neither i nor j fail in r' , 3) k has initial value 0 in r' , and 4) $Z_j = Z_i \cup \{k\}$ in r' . (Note that by definition, Z_i has the same value in both r and r' .) By the induction hypothesis (switching the roles of i and j), j decides v before or at time 1 at r' , and therefore by **Agreement**, i cannot decide $1-v$ in r' , and hence it does not decide $1-v$ up to time 1 in r .
- II. If there exists a process $k \neq i$ with initial value $1-v$, s.t. $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$, then $k \notin \{i, j\}$. Hence, as $t > 0$, there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) neither i nor j fail in r' , 3) $\langle k, 0 \rangle$ is hidden from $\langle j, 1 \rangle$ in r' , and 4) $Z_j = Z_i$ in r' . (Once again, Z_i has the same value in both r and r' .) By Case I (switching the roles of i and j), j decides v before or at time 1 in r' , and therefore by **Agreement**, i cannot decide $1-v$ in r' , and hence it does not decide $1-v$ up to time 1 in r .
- III. Otherwise, $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$ for all processes k , and k has initial value v for all processes $k \neq i$. As $|Z_i| < n$, we have that i has initial value $1-v$. Thus, there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) $f = 0$ in r' , and 3) $Z_j = Z_i$ in r' . (Once again, Z_i has the same value in both r and r' .) As i has initial value $1-v$ in r' as well, by Case II (switching the roles of i and j), j decides v before or at time 1 in r' , and therefore by **Agreement**, i cannot decide $1-v$ in r' , and hence it does not decide $1-v$ up to time 1 in r , and the proof is complete. \square

Lemma 15 (No Earlier Decisions). *Assume that $n > 2$ and $t > 0$. Let $Q \preceq \text{OPT}_{\text{Maj}}$ solve Consensus and let r be a run of Q . Let i be a process and let m be a time, s.t. $\neg K_i(\text{Maj} = 0)$ and $\neg K_i(\text{Maj} = 1)$. If there exists a hidden path w.r.t. $\langle i, m \rangle$, then i does not decide at (r, m) .*

Proof. Let $v \in \{0, 1\}$ be a value. We show that i does not decide v at (r, m) .

We first consider the case in which $m = 0$. In this case, there exists a run r' of Q s.t. 1) $r'_i(0) = r_i(0)$, 2) $\text{Maj} = 1-v$, and 3) $f = 0$. As $f = 0$ and $\text{Maj} = 1-v$ in r' , we have $K_i(\text{Maj} = 1-v)$

at $(r', 1)$, and therefore, by Lemma 14, i decides $1-v$ before or at 1 in r' ; therefore, i does not decide v at $(r', 0)$, and hence neither does it decide v at $(r, 0) = (r, m)$.

We turn to the case in which $m > 0$. As there exists a hidden path w.r.t. $\langle i, m \rangle$, for every $0 \leq \ell \leq m$ there exists a process b_ℓ s.t. $\langle b_\ell, \ell \rangle$ is hidden from $\langle i, m \rangle$. Thus, there exists a run r' of Q s.t. 1) $r'_i(m) = r_i(m)$, 2) $\text{Maj} = 1-v$, 3) $\langle b_1, 1 \rangle$ sees $\langle k, 0 \rangle$ for all processes k (and therefore $K_{b_1}(\text{Maj} = 1-v)$ at 1, 4) $\langle b_\ell, \ell \rangle$ is seen by $\langle b_{\ell+1}, \ell+1 \rangle$ for every $1 \leq \ell < m$, and 5) neither b_m nor i fail in r' . We show by induction that b_ℓ decides $1-v$ before or at ℓ in r' , for every $1 \leq \ell \leq m$.

Base: By Lemma 14, b_1 decides $1-v$ before or at 1 in r' .

Step: Let $1 < \ell \leq m$ and assume that $b_{\ell-1}$ decides $1-v$ before or at $\ell-1$ in r' . As $\langle b_{\ell-1}, \ell-1 \rangle$ is seen by $\langle b_\ell, \ell \rangle$ in r' , there exists a run $r'' = Q[\gamma]$ of Q , s.t. 1) $r''_{b_\ell}(\ell) = r'_{b_\ell}(\ell)$, and 2) Neither $b_{\ell-1}$ nor b_ℓ fail in r'' . As $\langle b_{\ell-1}, \ell-1 \rangle$ is seen by $\langle b_\ell, \ell \rangle$, and as $r''_{b_\ell}(\ell) = r'_{b_\ell}(\ell)$, $b_{\ell-1}$ decides $1-v$ before or at $\ell-1$ in r'' as well. As neither $b_{\ell-1}$ nor b_ℓ fail in r'' , by **Agreement** b_ℓ does not decide v before or at ℓ in r'' . As $\langle b_1, 1 \rangle$ is seen by $\langle b_\ell, \ell \rangle$ in r' , we have $K_{b_\ell}(\text{Maj} = 1-v)$ at (r', ℓ) , and therefore also at (r'', ℓ) . Thus, b_ℓ decides in $(\text{OPT}_{\text{Maj}}[\gamma], \ell)$, and therefore b_ℓ decides before or at ℓ in r'' , and so it decides $1-v$ before or at ℓ in r'' , and hence it also decides $1-v$ before or at ℓ in r' , and the proof by induction is complete.

As we have shown, b_m decides $1-v$ in r' . As neither b_m nor i fail in r' , by **Agreement** i does not decide v at (r', m) , and therefore neither does it decide v at (r, m) . \square

We can now prove Theorem 3.

Proof of Theorem 3. **Agreement**, **Decision** and **Validity** are straightforward and left to the reader. If $n > 2$, then unbeatability follows from Lemma 15. If $n = 1$, then it is straightforward to verify that the single process always decides at time 0, and so OPT_{Maj} cannot be improved upon. Finally, if $n = 2$, then it is easy to check that OPT_{Maj} is equivalent to OPT_0 , and so is unbeatable.

The fact that all decisions are performed by time $f + 1$ follows, exactly as in Lemma 7, from the fact that a hidden path exists w.r.t. each undecided process. \square

We note that the condition $t > 0$ in Theorem 3 cannot be dropped if $n > 2$. Indeed, if $t = 0$ and $n > 2$, then both OPT_0 and OPT_1 (in which some decisions are made at time 0, and the rest — at time 1) strictly dominate OPT_{Maj} (in which all decisions are made at time 1).

A.3 Uniform Consensus

We note that while the assumption $t < n$ simplifies presentation throughout the proofs below, the case $t = n$ can be analysed via similar tools.

Proof of Lemma 9. Let P be a uniform consensus protocol, and let r be a run of P such that $(R_P, r, m) \not\models K_i \exists \text{correct}(v)$. Thus, there exists a run $r' \in P[\alpha']$ such that $r_i(m) = r'_i(m)$ and $(R_P, r', m) \not\models \exists \text{correct}(v)$. Consider the adversary β that agrees with α' up to time m , and in which all active but faulty processes at (r', m) crash at time m without sending any messages. $\beta \in \gamma_{\text{cr}}^t$ because it has a legal input vector (identical to α'), and at most t crash failures, as it has the same set of faulty processes as $\alpha' \in \gamma_{\text{cr}}^t$. It follows that $r'' = P[\beta]$ is a run of P . Since β agrees with α' on the first m rounds, we have that $r''_i(m) = r'_i(m)$. Nonetheless, no correct process will ever know $\exists v$ in r'' , and thus by **Validity** no correct process ever decides v in r'' . By decision, all correct processes thus decide not on v . By **Uniform Agreement**, and as $t < n$ (i.e. there are correct processes), i cannot decide on v in r'' , and thus, as $r''_i(m) = r'_i(m) = r_i(m)$, it cannot decide on v in r at m . \square

Before moving on to prove Lemma 10. We first introduce some notation.

Definition 5. For a node $\langle i, m \rangle$, we denote by $F\langle i, m \rangle \in \{0, \dots, t\}$ the number of failures known to $\langle i, m \rangle$, i.e. the number of processes $j \neq i$ from which i does not receive a message at time m .

We note that \mathbf{d} , as defined in Lemma 10, is precisely $F\langle i, m \rangle$.

Proof of Lemma 10 (Sketch). It is straightforward to see that each of conditions (a) and (b) implies $K_i \exists \text{correct}(v)$ (Condition (a): as $\langle i, m-1 \rangle$ is seen at m by all correct processes; condition (b): as the number of distinct processes knowing $\exists 0$, including i itself, is greater than the maximum number of active processes that can yet fail). If neither condition holds, then i considers it possible that only incorrect processes know $\exists v$, and that they all immediately fail (i at time m before sending any messages, and the others — immediately after sending the last message seen by i), in which case no correct process would ever know $\exists v$. \square

As with P_0 in the case of consensus, by analysing decisions in protocols dominating $U-P_0$, we show that no Uniform Consensus protocol can dominate $U-\text{OPT}_0$. Lemmas 17 and 18 give sufficient conditions for deciding 0 in any Uniform Consensus protocol dominating $U-P_0$. As mentioned above, the analysis is considerably subtler for Uniform Consensus, because the analogue of Lemma 4 is not true. Receiving a message with value 0 in a protocol dominating $U-P_0$ does not imply that the sender has decided 0.

Lemma 16 (No decision at time 0). *Assume that $t > 0$. Let Q solve Uniform Consensus. No process decides at time 0 in any run of Q .*

Proof. As $t < n$, by Lemma 9 it is enough to show that $\neg K_i \exists v$ for every process i and $v \in \{0, 1\}$. As $0 < t$, and as $F\langle i, 0 \rangle = 0$ for all processes i by definition, we have that by Lemma 10, the proof is complete. \square

Lemma 17 (Decision at time 1). *Let $Q \preceq U-P_0$ solve Uniform Consensus and let $r = r[\alpha]$ be a run of Q . Let i be a process with initial value 0 in r s.t. i is active at time 1 in r . If either of the following hold in r , then $\langle i, 1 \rangle$ decides 0 in r .*

1. $t > 0$ and there exists a process $j \neq i$ with initial value 0 s.t. $\langle j, 0 \rangle$ is seen by $\langle i, 1 \rangle$.
2. $t > 1$ and $F\langle i, 1 \rangle < t$.

Proof. For both parts, we first note that by Lemma 10 and by definition of $U-P_0$, i decides 0 at $(U-P_0[\alpha], 1)$. As $Q \preceq U-P_0$, we thus have that i must decide upon some value in r by time 1. By Lemma 16, i does not decide at $(r, 0)$. Thus, i must decide at $(r, 1)$.

We now show Part 1 by induction on $n - |Z_i^0|$, where Z_i^0 is defined to be the set of processes k with initial value 0, s.t. $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$. Note that by definition, $i, j \in Z_i^0$, and so $1 < |Z_i^0| \leq n$.

Base: $|Z_i^0| = n$. In this case, all initial values are 0, and so by **Validity** i decides 0 at $(r, 1)$.

Step: Let $1 < \ell < n$ and assume that Part 1 holds whenever $|Z_i^0| = \ell + 1$. Assume that $|Z_i^0| = \ell$. We reason by cases.

- I. If there exists a process k s.t. $\langle k, 0 \rangle$ is hidden from $\langle i, 1 \rangle$, then there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) j is active at $(r', 1)$, 3) k has initial value 0 in r' , and 4) $Z_j^0 = Z_i^0 \cup \{k\}$ in r' . (Note that by definition, Z_i^0 has the same value in both r and r' .) By the induction hypothesis (switching the roles of i and j), j decides 0 at $(r', 1)$, and therefore by **Uniform Agreement**, i cannot decide 1 at $(r', 1)$, and hence it does not decide 1 at $(r, 1)$. Thus, i decides 0 at $(r, 1)$.

II. Otherwise, $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$ for all processes k . As $|Z_i^0| < n$, there exists a process $k \notin Z_i^0$ (in particular, $k \notin \{i, j\}$). Hence, as $t > 0$, there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) j is active at $(r', 1)$, 3) $\langle k, 0 \rangle$ is hidden from $\langle j, 1 \rangle$ in r' , and 4) $Z_j^0 = Z_i^0$ in r' . (Once again, Z_i^0 has the same value in both r and r' .) By Case I (switching the roles of i and j), j decides 0 at $(r', 1)$, and therefore by **Uniform Agreement**, i cannot decide 1 at $(r', 1)$, and hence it does not decide 1 at $(r, 1)$. Thus, i decides 0 at $(r, 1)$.

We move on to prove Part 2. If $\langle k, 0 \rangle$ is hidden from $\langle i, 1 \rangle$ for all processes $k \neq i$, then $\neg K_i \exists 1$ at $(r, 1)$. Thus, by Lemma 9, i cannot decide 1 at $(r, 1)$, and so must decide 0 at $(r, 1)$. Otherwise, there exists a process $k \neq i$ s.t. $\langle k, 0 \rangle$ is seen by $\langle i, 1 \rangle$. As $n > t > 1$, we have $n > 2$ and so there exists a process $j \notin \{i, k\}$; if $F\langle i, 1 \rangle > 0$, then we pick j s.t. $\langle j, 0 \rangle$ is hidden from $\langle i, 1 \rangle$. Since $t > 1$ (for the case in which $F\langle i, 1 \rangle = 0$ and $\langle j, 0 \rangle$ is seen by $\langle i, 1 \rangle$) and since $t > F\langle i, 1 \rangle$ (for the case in which $\langle j, 0 \rangle$ is hidden from $\langle i, 1 \rangle$), there exists a run r' of Q , s.t. 1) $r'_i(1) = r_i(1)$, 2) k never fails in r' , 3) j fails at $(r', 0)$ before sending any messages except perhaps to i , and 4) i fails at $(r', 1)$, immediately after deciding but before sending any messages. Thus, there exists a run r'' of Q , s.t. 1) $r''_k(m') = r'_k(m')$ for all m' , 2) k never fails in r'' , 3) i and j both have initial value 0 in r'' , 4) j fails at $(r'', 0)$ while successfully sending a message only to i (and therefore $j \in Z_i^0$ in r''), and 5) i fails at $(r'', 1)$, immediately after deciding but before sending out any messages. By Part 1, i decides 0 at $(r'', 1)$, and therefore k can never decide 1 during r'' , and therefore neither during r' . As k never fails during r' , by **Decision** it must thus decide 0 at some point during r' . Therefore, by **Uniform Agreement**, i cannot decide 1 at $(r', 1)$, and thus it does not decide 1 at $(r, 1)$. Thus, i decides 0 at $(r, 1)$. \square

Lemma 18 (Decision at times later than 1). *Let $Q \preceq_{U-P_0}$ solve Uniform Consensus, let $r = Q[\alpha]$ be a run of Q and let $m > 0$. Let i be a process s.t. $K_i \exists 0$ holds at time m for the first time in r , s.t. $K_i \exists \text{correct}(0)$ holds at time $m + 1$ for the first time in r , and s.t. i is active at $(r, m + 1)$. If either of the following hold in r , then i decides 0 at $(r, m + 1)$.*

1. *All of the following hold.*

- $F\langle i, m + 1 \rangle < t$.
- *There exists a process z s.t. $K_z \exists 0$ holds at time $m - 1$, s.t. $\langle z, m - 1 \rangle$ is seen by $\langle i, m \rangle$, but s.t. $\langle z, m \rangle$ is not seen by $\langle i, m + 1 \rangle$,*
- *There exists a process $j \neq i$ s.t. $\langle j, m \rangle$ is seen by $\langle i, m + 1 \rangle$ and $\langle z, m - 1 \rangle$ is seen by $\langle j, m \rangle$.*

2. $F\langle i, m + 1 \rangle < t - 1$.

Proof. We prove the lemma by induction on m , with the base and the step sharing the same proof (as will be seen below, the conceptual part of an induction base will be played, in a sense, by Lemma 17).

We prove both parts together, highlighting local differences in reasoning for the different parts as needed. For Part 2, we denote by z an arbitrary process s.t. $K_z \exists 0$ holds at time $m - 1$ and s.t. $\langle z, m - 1 \rangle$ is seen by $\langle i, m \rangle$. (As $m > 0$, such a process must exist for i to know $\exists 0$ at time m for the first time; nonetheless, unlike when proving Part 1, it is not guaranteed when proving this part that $\langle z, m \rangle$ is not seen by $\langle i, m + 1 \rangle$.)

We first note that by Lemma 9 and by definition of $U-P_0$, i decides 0 at $(U-P_0[\alpha], m + 1)$. As $Q \preceq_{U-P_0}$, we thus have that i must decide upon some value in r by time $m + 1$. By Lemma 9, the precondition for deciding 0 is not met by i at (r, m) . Therefore, it is enough to show that i does not decide 1 before or at time $m + 1$ in r in order to show that i decides 0 at $(r, m + 1)$.

Let $Z_i^{z,m}$ be the set of processes k s.t. $\langle k, m \rangle$ is seen by $\langle i, m+1 \rangle$ in r and s.t. $\langle z, m-1 \rangle$ is seen by $\langle k, m \rangle$ in r . (By definition, $i \in Z_i^{z,m}$.) Let C_i be the set of all processes k s.t. $\langle k, m \rangle$ is either seen by, or hidden from $\langle i, m+1 \rangle$ (i.e. the set of nodes that $\langle i, m+1 \rangle$ does not know to be inactive at time m). Note that by definition, $Z_i^{z,m} \subseteq C_i$. We first consider the case in which $Z_i^{z,m} \supsetneq \{i\}$, and prove the m -induction step (for the given m) for this case by induction on $|C_i \setminus Z_i^{z,m}|$.

Base: $Z_i^{z,m} = C_i$. In this case, $\langle i, m+1 \rangle$ does not know that z fails at time $m-1$. Thus, $z \in C_i$ and therefore $z \in Z_i^{z,m}$. It follows that $\langle z, m \rangle$ is seen by $\langle i, m+1 \rangle$ and therefore the second condition of Part 1 does not hold. Thus, the condition of Part 2 holds: $F\langle i, m+1 \rangle < t-1$. Furthermore, we thus have that z is active at time m . We now argue that z decides 0 at (r, m) , which completes the proof of the base case, as by **Uniform Agreement** i can never decide 1 during r . We reason by cases; for both cases, note that since $\langle z, m \rangle$ is seen by $\langle i, m+1 \rangle$, we have that $F\langle z, m \rangle \leq F\langle i, m+1 \rangle < t-1$.

- If $m = 1$: As $K_z \exists 0$ at time $m-1 = 0$, z has initial value 0. As $F\langle z, m \rangle < t-1$, we have that $t > 1$. By Part 2 of Lemma 17 (for $i = z$), we thus have that z decides 0 at $(r, 1) = (r, m)$.
- Otherwise, $m > 1$. In this case, as $\langle z, m-2 \rangle$ is seen by $\langle i, m-1 \rangle$, and as $K_z \exists 0$ holds at time m for the first time, we have that $K_z \exists 0$ holds at time $m-1$ for the first time. Similarly, as $\langle z, m-1 \rangle$ is seen by $\langle i, m \rangle$, and as $K_z \exists \text{correct}(0)$ does not hold at time m , we have that $K_z \exists \text{correct}(0)$ does not hold at time $m-1$. By Part 2 of the m -induction hypothesis (for $i = z$), z decides 0 at (r, m) .

Step: Let $\{i\} \subsetneq Z_i^{z,m} \subsetneq C_i$, and assume that the claim holds whenever $Z_i^{z,m}$ is of larger size. For Part 1, note that $j \in Z_i^{z,m}$, for j as defined in the conditions for that part; for Part 2, let $j \in Z_i^{z,m}$ be arbitrary. Analogously to the proof of the induction step in the proof of Part 1 of Lemma 17, we reason by cases. For the time being, assume that the conditions of Part 2 hold, i.e. that $F\langle i, m+1 \rangle < t-1$.

- I. If there exists a process $k \in C_i$ s.t. $\langle k, m \rangle$ is hidden from $\langle i, m+1 \rangle$, then there exists a run r' of Q , s.t. 1) $r'_i(m+1) = r_i(m+1)$, 2) j is active at $(r', m+1)$, 3) $\langle z, m-1 \rangle$ is seen by $\langle k, m \rangle$ in r' , and 4) $Z_j^{z,m} = Z_i^{z,m} \cup \{k\}$ and $C_j = C_i$ in r' . (Note that by definition, $Z_i^{z,m}$ and C_i have the same values in both r and r' .) We note that $F\langle j, m+1 \rangle = F\langle i, m+1 \rangle - 1$ in r' , and that by definition $F\langle i, m+1 \rangle$ is the same in both r and r' . By the inductive hypothesis for $Z_j^{z,m}$ (i.e., for j w.r.t. z at time m), j decides 0 at $(r', m+1)$, and therefore by **Uniform Agreement**, i cannot decide 1 in r' , and therefore it cannot decide 1 before or at $m+1$ in r' , and the proof is complete.
- II. Otherwise, for each process $k \in C_i$, $\langle k, m \rangle$ is seen by $\langle i, m+1 \rangle$. As $Z_i^{z,m} \subsetneq C_i$, there exists a process $k \neq i$ s.t. $\langle k, m \rangle$ is seen by $\langle i, m+1 \rangle$ but s.t. $\langle z, m-1 \rangle$ is hidden from $\langle k, m \rangle$ (thus $k \neq j$). Hence, and since $F\langle i, m+1 \rangle < t$, there exists a run r' of Q , s.t. 1) $r'_i(m+1) = r_i(m+1)$, 2) j is active at $(r', m+1)$, 3) $\langle k, m \rangle$ is hidden from $\langle j, m+1 \rangle$ in r' , and 4) $Z_j^{z,m} = Z_i^{z,m}$ and $C_j \supseteq C_i$ in r' . (Once again, $Z_i^{z,m}$ and C_i have the same values in both r and r' .) We note that $F\langle j, m+1 \rangle = F\langle i, m+1 \rangle + 1$ in r' , and that once more, by definition, $F\langle i, m+1 \rangle$ is the same in both r and r' . By Case I (for $i = j$), and since Case I uses the inductive hypothesis for $Z_j^{z,m}$ with one less failure, we conclude that j decides 0 at $(r', m+1)$. Therefor, by **Uniform Agreement**, i cannot decide 1 at $(r', m+1)$, and thus it cannot decide 1 before or at $m+1$ in r , and the proof is complete.

To show that the $Z_i^{z,m}$ -induction step also holds under the conditions of Part 1, we observe that since $\langle z, m \rangle$ is not seen by $\langle i, m+1 \rangle$ in this case, the amount of invocations of Case II (which

uses Case I with one additional known failure) before reaching the $Z_i^{z,m}$ -induction base is strictly smaller than that of Case I (which uses the $Z_i^{z,m}$ -induction hypothesis with one less known failure), and therefore the $Z_i^{z,m}$ -induction base is reached with less known failures, i.e. with less than $t - 1$ known failures, i.e. the conditions of Part 2 hold at that point.

Finally, we consider the case in which $Z_i^{z,m} = \{i\}$. As any j as in Part 1 satisfies $j \in Z_i^{z,m}$, we have that the conditions of Part 2 hold, i.e. $F\langle i, m+1 \rangle < t-1$. Furthermore, we have that $\langle z, m \rangle$ is not seen by $\langle i, m+1 \rangle$ (otherwise, $z \in Z_i^{z,m}$). As $F\langle i, m+1 \rangle < t-1 < n-2$, there exist two distinct processes $j, k \neq i$ that are not known to $\langle i, m+1 \rangle$ to fail (and thus i, j, k, z are distinct). Thus, $\langle j, m \rangle$ and $\langle k, m \rangle$ are seen by $\langle i, m+1 \rangle$.

By definition of j, k , there exists a run r' of Q , s.t. 1) $r'_i(m+1) = r_i(m+1)$, 2) k never fails in r' , 3) j fails at (r', m) before sending any messages, 4) i fails at $(r', m+1)$, immediately after deciding but before sending any messages, and 5) the faulty processes in r' are those known by $\langle i, m \rangle$ to fail in r , and in addition i and j . We note that by definition, $F\langle i, m+1 \rangle$ is the same in r and r' , even though the number of failures in r' is $F\langle i, m+1 \rangle + 2$. We notice that there exists a run r'' of Q , s.t. 1) $r''_k(m') = r''_k(m')$ for all m' , 2) k never fails in r'' , 3) $\langle z, m-1 \rangle$ is seen by both $\langle i, m \rangle$ and $\langle j, m \rangle$ in r'' , 4) j fails at (r'', m) while successfully sending a message only to i (and therefore both $j \in Z_i^{z,m}$ and $F\langle i, m+1 \rangle < t-1$ in r''), and 5) i fails at $(r'', m+1)$, immediately after deciding but before sending out any messages. By the proof for the case in which $Z_i^{z,m} \supsetneq \{i\}$ ($j \in Z_i^{z,m}$), i decides 0 at $(r'', m+1)$, and therefore k can never decide 0 during r'' , and therefore neither during r' . As k never fails during r' , by **Decision** it must thus decide 0 at some point during r' . Therefore, by **Uniform Agreement**, i cannot decide 1 before or at $m+1$ in r' , and thus it does not decide 1 before or at $m+1$ in r , and the proof is complete. \square

Now that we have established when processes must decide 0 in any protocol dominating P_0 , we can deduce when processes cannot decide in any such protocol.

Lemma 19 (No Earlier Decisions when $K_i \exists 0$). *Let $Q \preceq \cup\text{-}P_0$ solve Uniform Consensus, let r be a run of Q , let m be a time, and let i be a process. If at time m in r we have $K_i \exists 0$, but $\neg K_i \exists \text{correct}(0)$, then i does not decide at (r, m) .*

Proof. If $m=0$, then by Lemma 10 and since $\neg K_i \exists \text{correct}(0)$ at $m=0$ (even though $K_i \exists 0$), we have $t > 0$. Thus, by Lemma 16, i does not decide at (r, m) . Assume henceforth, therefore, that $m > 0$.

As $\neg K_i \exists \text{correct}(0)$, we have that by Lemma 10, $\neg K_i \exists 0$ at time $m-1$. Thus, there exists a process z s.t. $K_z \exists 0$ at $m-1$, and $\langle z, m-1 \rangle$ is seen by $\langle i, m \rangle$. In turn, by Lemma 10, we have that $F\langle i, m \rangle < t-1$. There exists a run r' of Q , s.t. 1) $r'_i(m) = r_i(m)$, and 2) the faulty processes in r' are those known by $\langle i, m \rangle$ to fail in r . We henceforth reason about r' . By definition of r' , $F\langle i, m+1 \rangle = F\langle i, m \rangle < t-1$ (by definition, the value of $F\langle i, m \rangle$ is the same in both r and r'). Thus, by Part 2 of Lemma 18, i decides 0 at $(r', m+1)$, and hence i does not decide at (r', m) , and therefore neither does it decide at (r, m) . \square

Lemma 20 (No Earlier Decisions when $\neg K_i \exists 0$). *Assume that $t > 0$. Let $Q \preceq \cup\text{-}P_0$ solve Uniform Consensus, let r be a run of Q , let m be a time, and let i be a process. If there exists a hidden path w.r.t. $\langle i, m \rangle$ in r , and if at time m in r we have $\neg K_i \exists 0$, then i does not decide at (r, m) .*

Proof. As $\neg K_i \exists 0$ at time m , then by **Validity**, i does not decide 0 at (r, m) . Thus, it is enough to show that i does not decide 1 at (r, m) in order to complete the proof. If $m=0$, then by Lemma 16, i does not decide 1 at (r, m) either. Assume henceforth, therefore, that $m > 0$.

As there exists a hidden path w.r.t. $\langle i, m \rangle$, there exist processes $z, j \neq i$ s.t. $\langle z, m-1 \rangle$ is hidden from $\langle i, m \rangle$ and s.t. $\langle j, m-1 \rangle$ is seen by $\langle i, m \rangle$.

We first consider the case in which $F\langle i, m \rangle < t$. In this case, there exists a run $r' = Q[\beta]$ of Q , s.t. all of the following hold in r' :

- $r'_i(m) = r_i(m)$.
- z is the unique process that knows $\exists 0$ at $m-1$, and knows so then for the first time, either having initial value 0 (if $m=1$) or (as explained in the Nonuniform Consensus section) seeing only a single node that knows $\exists 0$ at $m-2$ (if $m>1$).
- z fails at $(r', m-1)$, successfully sending messages to all nodes except for i .
- The faulty processes in r' are those known by $\langle i, m \rangle$ to fail in r , and in addition i , which fails at time m without sending out any messages. In particular, j never fails.

We henceforth reason about r' . First, we note that $\langle j, m+1 \rangle$ does not know that z fails at $m-1$ (as opposed to at m). As $\langle j, m \rangle$ sees $\langle z, m-1 \rangle$, as $K_z \exists 0$ at $m-1$, and as j never fails, by Lemma 10 we have that $K_j \exists \text{correct}(0)$ at $(r', m+1)$. Thus, j decides at $(U-P_0[\beta], m+1)$, and so j must decide before or at $m+1$ in r' . As $r_i(m) = r'_i(m)$, then by **Uniform Agreement** it is enough to show that j does not decide 1 up to time $m+1$ in r' in order to complete the proof.

There exists a run r'' of Q , s.t. 1) $r''_j(m+1) = r'_j(m+1)$, and 2) the only difference between r'' and r' up to time m is that in r'' , z fails only at time m , after deciding but without sending a message to j . By **Uniform Agreement**, it is enough to show that z decides 0 at (r'', m) in order to complete the proof.

We henceforth reason about r'' . As z does not know at m that neither z nor i fail, we have $F\langle z, m-1 \rangle \leq F\langle z, m \rangle < t-1$. Thus, $t > 1$. If $m=1$, we therefore have by Part 2 of Lemma 17 that z decides 0 at (r'', m) . Otherwise, $m > 1$. As $K_z \exists 0$ at $m-1$ for the first time, as $\langle z, m-1 \rangle$ sees only one node at $m-1$ that knows $\exists 0$, and as $F\langle z, m \rangle < t-1$, by Lemma 10 we have $\neg K_z \exists \text{correct}(0)$ at $m-1$. Thus, by Part 2 of Lemma 18 (for $i = z$), z decides 0 at (r'', m) . Either way, the proof is complete.

We now consider the case in which $F\langle i, m \rangle = t$. There exists a run $r' = Q[\beta]$ of Q , s.t. all of the following hold:

- $r'_i(m) = r_i(m)$.
- All processes k s.t. $\langle k, m-1 \rangle$ is hidden from $\langle i, m \rangle$ (including $k = z$) know $\exists 0$ at $(r', m-1)$, either having initial value 0 (if $m=1$) or all seeing only a single node that knows $\exists 0$ at $m-2$ (and which fails at time $m-2$ without being seen by $\langle i, m \rangle$) — denote this node by z' .
- All such processes fail at time $m-1$, successfully sending messages to all nodes except for i .
- The faulty processes failing in r' are those known by $\langle i, m \rangle$ to fail in r . In particular, there are t such processes.

We henceforth reason about r' . We note that as i never fails, $F\langle i, m-1 \rangle \leq F\langle j, m \rangle$ (equality can actually be shown to hold here, but we do not need it). As the number of nodes at $m-1$ knowing $\exists 0$ that are seen by $\langle j, m \rangle$ equals $F\langle i, m \rangle - F\langle i, m-1 \rangle \geq t - F\langle j, m \rangle$ (by the above remark, equality holds here as well), we have by Lemma 10 that $K_j \exists \text{correct}(0)$ at m , and therefore j decides at $(U-P_0[\beta], m)$; thus, it must decide before or at m in r' . As $r_i(m) = r'_i(m)$, by **Uniform Agreement** it is enough to show that j does not decide 1 up to time m in r' in order to complete the proof.

We proceed with an argument similar in a sense to those of Part 1 of Lemma 17 and the inner induction in the proof of Lemma 18.

As $\langle z, m-1 \rangle$ is seen by $\langle j, m \rangle$, there exists a run r'' of Q , s.t. 1) $r''_j(m) = r'_j(m)$, and 2) the only difference between r'' and r' up to time m is that in r' , z never fails, but rather i fails at $m-1$ after sending a message to j but without sending a message to z . We note that there are t processes failing throughout r'' . We henceforth reason about r'' . If $m=1$, then z has initial value 0 and if $m > 1$, then $\langle z, m-1 \rangle$ sees $\langle z', m-2 \rangle$; either way, by Lemma 10, $K_z \exists \text{correct}(0)$ at (r'', m) and therefore z must decide before or at time m . Thus, it is enough to show that z does not decide 1 up to time m in r'' in order to complete the proof.

As $\langle i, m-1 \rangle$ is not seen by $\langle z, m \rangle$, there exists a run r''' of Q , s.t. 1) $r'''_z(m) = r''_z(m)$, and 2) the only difference between r''' and r'' up to time m is that in r''' , $\langle i, m-1 \rangle$ sees $\langle z', m-2 \rangle$ (or, if $m = 1$, then the difference is that i has initial value 0); we note that $\langle i, m-1 \rangle$ is still seen by $\langle j, m \rangle$. We note that there are t processes failing throughout r''' . Observe that the number of nodes at $m-1$ knowing $\exists 0$ that are seen by $\langle j, m \rangle$ in r''' is greater than in r'/r'' (between which j at m cannot distinguish), however $F\langle j, m \rangle$ remains the same between r'/r'' and r''' ; thus, $K_j \exists \text{correct}(0)$ at m in r''' as well, and therefore j must decide before or at time m in r''' . Thus, it is enough to show that j does not decide 1 up to time m in r''' in order to complete the proof. We henceforth reason about r''' .

As $\langle i, m-1 \rangle$ is seen by $\langle j, m \rangle$, there exists a run r'''' of Q , s.t. 1) $r''''_j(m) = r'''_j(m)$, and 2) the only difference between r'''' and r''' up to time m is that in r'''' , i does not fail (and is thus seen by $\langle z, m \rangle$). We note that there are $t-1$ processes failing throughout r'''' , and thus in particular $F\langle z, m \rangle < t$. If $m = 1$, then by Part 1 of Lemma 17 (for $i = z$ and $j = i$), z decides 0 in (r'''', m) . Otherwise, i.e. if $m > 1$, by Part 1 of Lemma 18 (for $i = z$, $z = z'$, and $j = i$), z decides 0 in (r'''', m) . Either way, the proof is complete. \square

From Lemmas 19 and 20, we deduce sufficient conditions for unbeatability of Uniform Consensus protocols dominating $U-P_0$; these conditions also become necessary if it can be shown that there exists some Uniform Consensus protocol dominating $U-P_0$ that meets them, as we indeed show momentarily for $U-\text{OPT}_0$.

Lemma 21. *Assume that $0 < t < n$. A protocol $Q \preceq U-P_0$ that solves Uniform Consensus and in which a node $\langle i, m \rangle$ decides whenever any of the following hold at m , is an unbeatable Uniform Consensus protocol.*

- $K_i \exists \text{correct}(0)$.
- No hidden path w.r.t. $\langle i, m \rangle$ exists, and $\neg K_i \exists 0$.

Proof. Directly from Lemmas 19 and 20. \square

By Lemma 21, we have that if $U-\text{OPT}_0$ solves Uniform Consensus, then it does so in an unbeatable fashion.

Lemma 22. $U-\text{OPT}_0 \preceq U-P_0$

Proof. As explained above, at time $t+1$ no hidden paths exist (see the proofs of Lemma 7 and Theorem 3), and furthermore, by Lemma 3 we have at time $t+1$ that $K_i \exists 0$ iff $K_i \exists \text{correct}(0)$. The claim therefore holds by definition of $U-\text{OPT}_0$ and $U-P_0$. \square

Theorem 7. $U-\text{OPT}_0$ solves Uniform Consensus in γ_{cr}^t . Furthermore,

- If $f \geq t-1$, then all decisions are made by time $f+1$ at the latest.
- Otherwise, all decisions are made by time $f+2$ at the latest.

Proof. Decision: In some run of U-OPT_0 , let i be a process and let m be a time s.t. i is active at m but has not decided until m , inclusive. Let $\tilde{m} \leq m$ be the latest time not later than m s.t. a hidden path exists w.r.t. $\langle i, \tilde{m} \rangle$. We claim that as i is undecided at m , we have $\tilde{m} \geq m - 1$; indeed, otherwise, by i being undecided at $\tilde{m} + 1$ despite the absence of a hidden path w.r.t. $\langle i, \tilde{m} + 1 \rangle$, we would have $K_i \exists 0$ at $\tilde{m} + 1$, and so, by Lemma 10, we would have $K_i \exists \text{correct}(0)$ at $\tilde{m} + 2 \leq m$ — a contradiction to i being undecided at m .

As a hidden path exists w.r.t. $\langle i, \tilde{m} \rangle$, we have, as in the proofs of Lemma 7 and Theorem 3, that $\tilde{m} \leq f$; in fact, the same proof shows the even stronger claim $\tilde{m} \leq F\langle i, \tilde{m} \rangle$ — we will later return to this inequality. As $\tilde{m} \leq f$, we therefore have that $m \leq \tilde{m} + 1 \leq f + 1$. We thus have that every process that is active at time $f + 2$, decides by this time at the latest.

Before moving on to show **Validity** and **Uniform Agreement**, we first complete the analysis of stopping times. Assume that $m = f + 1$. (i is still a process that is active but undecided at m .) As $f = m - 1 \leq \tilde{m} \leq F\langle i, \tilde{m} \rangle \leq F\langle i, m \rangle \leq f$, we have that both $\tilde{m} = m - 1$ and $F\langle i, m \rangle = f$. As $\tilde{m} = m - 1$, we have that no hidden path exists w.r.t. $\langle i, m \rangle$. As i is undecided at m , we thus have, by definition of U-OPT_0 , that $K_i \exists 0$ while $\neg K_i \exists \text{correct}(v)$ at m . We therefore have that $K_i \exists 0$ at m for the first time. Therefore, as $m > \tilde{m} \geq 0$, there exists a process j such that $K_j \exists 0$ at $m - 1$ and s.t. $\langle j, m - 1 \rangle$ is seen by $\langle i, m \rangle$. Thus, by Lemma 10 and since $\neg K_i \exists \text{correct}(v)$, we have $F\langle i, m \rangle < t - 1$, and so $f = F\langle i, m \rangle < t - 1$.

We thus have that if $f = t - 1$, then every process that is active at time $f + 1$ decides by this time at the latest.

We move on to show **Validity** and **Uniform Agreement**. Henceforth, let i be a (possibly faulty) process that decides in some run of U-OPT_0 , let m be the decision time of i , and let v be the value upon which i decides.

Validity: If $v = 0$, then by definition $K_i \exists \text{correct}(0)$ at m , and so $K_i \exists 0$ at m , and in particular $\exists 0$. If $v = 1$, then by definition $\neg K_i \exists 0$, and so the initial value of i is 1, and so $\exists 1$. Either way, we have $\exists v$ as required.

Uniform Agreement: It is enough to show that if $v = 1$, then 0 is never decided upon in the current run. For the rest of this proof we assume, therefore, that $v = 1$; therefore, by definition of U-OPT_0 , we have that both $\neg K_i \exists 0$ and no hidden path exists w.r.t. $\langle i, m \rangle$. By Lemma 6, we therefore have that $K_i \text{not-known}(\exists 0)$ at m , and in particular $\text{not-known}(\exists 0)$ at m . By induction, as in the proof of Theorem 6, we have that $\text{not-known}(\exists 0)$ at every time later than m . In particular, we have that no correct process ever learns of an initial value of 0 (as $\text{not-known}(\exists 0)$ would never hold from that point on), and so $\exists \text{correct}(0)$ never holds; therefore, $K_j \exists \text{correct}(0)$ never holds for any j , and so by definition of U-OPT_0 no process ever decides upon 0, and the proof is complete. \square

Proof of Theorem 4. The claim follows from Lemma 21 and Theorem 7; in the boundary case of $t = 0$ (which is not covered by Lemma 21), we note that U-OPT_0 and OPT_0 coincide, as do the problems of uniform consensus and consensus; hence U-OPT_0 is unbeatable, and Theorem 4 holds, in that case as well. \square

Proof of Lemma 11. The proof has a similar structure to that of Lemma 8. opt-EDAUC decides either one round after the sender set repeats, or at time $t + 1$. As argued in the proof of Lemma 8, when the sender set repeats there is a round k all of whose nodes are revealed. If they don't contain evidence of an initial value of 0, then U-OPT_0 decides immediately. Otherwise, by Lemma 10(a) a correct process will know $\exists \text{correct}(0)$ and decide one round later, and if this occurs at time $m = t + 1$, then by Lemma 10(b) it will decide immediately. An adversary β on which U-OPT_0 beats opt-EDAUC with the claimed margins is a simplified version of the adversary α defined in the proof of Lemma 8. Denote the processes by $\text{Procs} = \{1, 2, \dots, n\}$. All initial values in β are 0. In round 1,

two processes crash—process 1 and process 2, with process 1 sending only to process n and nobody else, and process 2 sending to everyone except process n . No process fails in round 2, and in each of the rounds $m = 3, \dots, t$, process m crashes without sending any messages. Since precisely t processes fail in β we have that $\beta \in \text{Crash}(t)$. For $3 \leq m \leq t$, every correct process fails to hear from process m in round m for the first time. Every correct process $i \neq n$ fails to hear from process 1 in round 1 and from process 2 in round 2, while process n fails to hear from 2 in round 1 and from process 1 in round 2. In the protocol opt-EDAUC of [2], no process decides before its sender set repeats, and thus all decisions are taken at time $t + 1$ when the adversary is β . In U-OPT_0 , every correct process i sees $n - 1 \geq t + 1$ values of 0 in the first round. By Lemma 10(b) it follows that $K_i \exists \text{correct}(0)$ holds at time 1, the rule for $\text{decide}_i(0)$ in U-OPT_0 is satisfied, and process i decides 0 at time 1. \square

A.4 Efficient Implementation of Full-Information Protocols

We now sketch the structure of communication-efficient implementations for the protocols proposed in the paper:

Lemma 23. *For each of the protocols OPT_0 , OPT_{Maj} , and U-OPT_0 there is a protocol with identical decision times for all adversaries, in which every process sends at most $O(f \log n)$ bits overall to each other process.*

Proof (Sketch). Moses and Tuttle in [18] show how to implement full-information protocols in the crash failure model with linear-size messages. In our case, a further improvement is possible, since decisions in all of the protocols depend only on the identity of hidden nodes and on the vector of initial values. In a straightforward implementation, we can have a process i report “ $\text{value}(j) = v$ ” once for every j whose initial value it discovers, and “ $\text{failed_at}(j) = \ell$ ” once where ℓ is the earliest failure round it knows for j . In addition, it should send an “ I'm_alive ” message in every round in which it has nothing to report. Process i can send at most one value message and two failed_at messages for every j . Since I'm_alive is a constant-size message sent fewer than $f + 2$ times, and since encoding j ’s ID along with a failure round number $m \leq f + 2$ requires $\log n$ bits, a process i sends a total of $O(f \log n)$ bits overall. \square

A.5 Different Types of Unbeatability

We first formally define last-decider unbeatability.

Definition 6 (Last-Decider Domination and Unbeatability).

- A decision protocol Q **last-decider dominates** a protocol P in γ , denoted by $Q \preceq_{\gamma}^{l.d.} P$ if, for all adversaries α , if i the last decision in $P[\alpha]$ is at time m_i , then all decisions in $Q[\alpha]$ are taken before or at m_i . Moreover, we say that Q **strictly last-decider dominates** P if $Q \preceq_{\gamma}^{l.d.} P$ and $P \not\preceq_{\gamma}^{l.d.} Q$. I.e., if for some $\alpha \in \gamma$ the last decision in $Q[\alpha]$ is strictly before the last decision in $P[\alpha]$.
- A protocol P is a **last-decider unbeatable** solution to a decision task S in a context γ if P solves S in γ and no protocol Q solving S in γ strictly last-decider dominates P .

Remark 1.

- If $Q \preceq_{\gamma} P$, then $Q \preceq_{\gamma}^{l.d.} P$. (But not the other way around.)

- None of the above forms of strict domination implies the other.
- None of the above forms of unbeatability implies the other.

Last-decider domination does not imply domination in the sense of the rest of this paper (on which our proofs is based). Nonetheless, the specific property of protocols dominating OPT_0 , OPT_{Maj} , and U-OPT_0 , which we use to prove that these protocols are unbeatable, holds also for protocols that only last-decider dominate these protocols.

Lemma 24.

1. Let $Q \stackrel{\text{l.d.}}{\preceq} P_0$ satisfy **Decision**. If $K_i \exists 0$ at m in a run $r = Q[\alpha]$ of Q , then i decides in r no later than at m .
2. Let $Q \stackrel{\text{l.d.}}{\preceq} \text{OPT}_{\text{Maj}}$ satisfy **Decision**. If $K_i(\text{Maj} = v)$ for $v \in \{0, 1\}$ at m in a run $r = Q[\alpha]$ of Q , then i decides in r no later than at m .
3. Let $Q \stackrel{\text{l.d.}}{\preceq} \text{U-}P_0$ satisfy **Decision**. If $K_i \exists \text{correct}(0)$ at m in a run $r = Q[\alpha]$ of Q , then i decides in r no later than at m .

The main idea in the proof of each of the parts of Lemma 24 is to show that i considers it possible that all other active processes also know the fact stated in that part, and so they must all decide by the current time in the corresponding run of the dominated protocol. Hence, the last decision in that run is made in the current time; thus, by last-decider domination, i must decide. The proofs for the first two parts are somewhat easier, as in each of these parts, any process at m that sees (at least) the nodes seen by $\langle i, m \rangle$ (or has the same initial value, if $m=0$) also knows the relevant fact stated in that part. We demonstrate this by proving Part 1; the analogous proof of Part 2 is left to the reader.

Proof of Part 1 of Lemma 24. If $m = 0$, then there exists a run $r' = Q[\beta]$ of Q , s.t. 1) $r'_i(0) = r_i(0)$, 2) in r' all initial values are 0, and 3) i never fails in r' . Hence, in $P_0[\beta]$ all decisions are taken at time $m=0$, and therefore so is the last decision. Therefore, the last decision in r' must be taken at time 0. As i never fails in r' , by **Decision** it must decide at some point during this run, and therefore must decide at 0 in r' . As $r_i(0) = r'_i(0)$, i decides at 0 in r as well, as required.

If $m > 0$, then there exists a process j s.t. $K_j \exists 0$ at $m-1$ in r and $\langle j, m-1 \rangle$ is seen by $\langle i, m \rangle$. Thus, there exists a run $r' = Q[\beta]$ of Q , s.t. 1) $r'_i(m) = r_i(m)$, and 2) i and j never fail in r' . Thus, all processes that are active at m in r' see $\langle j, m-1 \rangle$ in r' and therefore know $\exists 0$ in r' . Hence, in $P_0[\beta]$ all decisions are taken by time m , and therefore so is the last decision. Therefore, the last decision in r' must be taken no later than at time m . As i never fails in r' , by **Decision** it must decide at some point during this run, and therefore must decide by m in r' . As $r_i(m) = r'_i(m)$, i decides by m in r as well, as required. \square

As the proof of Part 3 is slightly more involved, we show it as well.

Proof of Part 3 of Lemma 24. If $m = 0$, then by Lemma 10, $t = 0$. There exists a run $r' = Q[\beta]$ of Q , s.t. 1) $r'_i(0) = r_i(0)$, and 2) in r' all initial values are 0. Therefore, as $t = 0$, we have by Lemma 10 that all processes know $\exists \text{correct}(0)$ at $m = 0$ in r' . Hence, in $\text{U-}P_0[\beta]$ all decisions are taken at time $m = 0$, and therefore so is the last decision. Therefore, the last decision in r' must be taken at time 0 as well. Since $t = 0$, i never fails in r' , and so by **Decision** it must decide at some point during this run, and therefore must decide at 0 in r' . As $r_i(0) = r'_i(0)$, i decides at 0 in r as well, as required.

If $m > 0$, then there exists a process j s.t. $K_j \exists 0$ at $m-1$ in r and $\langle j, m-1 \rangle$ is seen by $\langle i, m \rangle$ in r . Furthermore, as $t < n$, there exists a set of processes I s.t. 1) $i, j \notin I$, 2) $|I| = t - F\langle i, m \rangle - 1$, and 3) $\langle k, m-1 \rangle$ is seen by $\langle i, m \rangle$ for every $k \in I$. Thus, there exists a run $r' = Q[\beta]$ of Q , s.t. 1) $r'_i(m) = r_i(m)$, 2) i and j never fail in r' , 3) all of I fail in r' at $m-1$, successfully sending messages only to i , and 4) every process at $m-1$ in r' that is not seen by $\langle i, m \rangle$, is not seen by any other process at m as well. We henceforth reason about r' . Every process $k \neq j$ that is active at m sees $\langle j, m-1 \rangle$ and furthermore satisfies $F\langle k, m \rangle \geq F\langle i, m \rangle + |I| = t - 1$. Thus, by Lemma 10, $K_k \exists \text{correct}(0)$ at m , and thus k decides at $(\text{U-}P_0[\beta], m)$. Additionally, as $K_j \exists 0$ at $m-1$, by Lemma 10 $K_j \exists \text{correct}(0)$ at m , and thus j decides at $(\text{U-}P_0[\beta], m)$. Hence, in $\text{U-}P_0[\beta]$ all decisions are taken by time m , and therefore so is the last decision. Therefore, the last decision in r' must be taken no later than at time m . As i never fails in r' , by **Decision** it must decide at some point during this run, and therefore must decide by m in r' . As $r_i(m) = r'_i(m)$, i decides by m in r as well, as required. \square

Proof of Theorem 5. As explained above, Theorem 5 follows from Lemma 24, and from the proofs of Theorems 2 to 4. \square